博士論文

模倣の重要性に着目した初学者向け プログラミング教育の研究

京都大学大学院情報学研究科
社会情報学専攻

岡本 雅子

平成 26 年 (2014 年)3 月

模倣の重要性に着目した初学者向け プログラミング教育の研究

岡本 雅子

内容梗概

プログラミングは，情報技術の基本的技能であり，現代の中等，高等の専門教育では情報技術者の育成のために初学者へのプログラミング教育が数多く実践されている．さらに，IT 業界の著名人がプログラミングを身に付けることを社会に訴える活動やオンラインでプログラミングを学習する環境を提供する企業などの動きもある．また，プログラミング教育は，専門教育に留まらず，普通教育でも取り入れられてきており，わが国の新しい学習指導要領では中学校の技術・家庭科において従来は選択であったプログラミングは，生徒全員が学ぶ事項になった．

これに対して，プログラミング教育の初学段階を対象とした研究は数多くあるし，様々な改善も試みられてはいる．しかしながら，その教授法について標準的な指針が示されるまでには至っておらず，さらなる研究と改善が必要とされているというのが現状である．本研究は，こうしたプログラミング教育に係る研究として，とくにその演習過程に着目し，その改善を試みたものである．

演習形態での授業は，当該授業の期間内に学習課題を達成することを目標とするという点において，知識伝達型や座学形態の授業と相違はないが，学習者自身が作業を進め，また，その作業から能動的に学ぶという学習形態であるため，教材そのものの使いやすさだったり，分かりやすさが直接的に影響する場面が散見される．例えば，学習の初期段階では，サンプルプログラムを写して実行するという模倣の過程を通して文法やコードの書き方を学ぶことが広く行われているが，ここでは，タイプミスに起因する文法エラーや実行時のエラーを多発する学習者が見られ，エラーを修正できないために学習そのものが停滞してしまう場合もある．このように，当該学習課題そのものではなく，プログラミングの作業を遂行していく過程に問題を抱える場合が少なからず見られる．また，作業はできても，これを介して学習課題を理解することにはつながらない場合も見られる．こうした点から，演習を授業として成立させるためには，教材やカリキュラムなどへの配慮など，自立的な学習の遂行やそれを介した課題

の理解への支援が重要であるといえる．

これに対して，著者は，プログラミング学習のなかで，サンプルプログラムを写して実行する模倣の過程を「写経型学習」と位置付け，この過程を充実，強化させること，また，その効果的な実施を目的として，初学段階における演習過程を対象とした教材のあり方について検討したいと考えた．

本研究では，まず，企業における初級プログラマの養成過程を調査し，写経型学習過程における学習者のつまずきや問題点など，初学者に見られる学習者の困難性（潜在的なつまずきの要因）を抽出した．

さらに，認知科学的視点から，認知的負荷理論に依拠して，この結果を
(1) 自立的な作業遂行において非本質的な認知的負荷を伴う場合
(2) 作業を介した理解において具体例の認知に困難性を伴う場合
(3) 作業を介した理解において当該学習課題それ自体の認知に困難性を伴う場合
の３つに類型化するとともにその類型に対応した学習方略について検討した．

つぎに，こうして得られた知見をもとに，写経型学習教材を開発し，最初に「自立的な作業遂行において非本質的な認知的負荷を伴う場合」のつまずきの軽減を試みたところ，手順に係るつまずきが大幅に減少しただけでなく，とくに初学者が典型的に困難を抱えるとされる繰り返し処理の理解を問うテストにおいて誤答者数の大幅な減少が見られるなど，当該学習課題そのものの理解についても大幅に改善されるといった結果が得られた．こうした結果は，写経型学習における作業としての模倣が滞りなく進むだけで，学習課題の理解も改善されるということを示しており，すなわちこれは，写経型学習そのものの重要性とその学習効果を示唆している．

また,「作業を介した理解において具体例の認知に困難性を伴う場合（非本質的な認知的負荷）」の改善を主な目的として,「視覚的顕在化」という方針をもとにマイコンボード教材を活用したカリキュラムを開発し，同教材を用いた実践を行ったところ，視覚的認知に係る側面において効果があったと推察できる結果が得られた．なお，この実践では,「作業を介した理解において当該学習課題それ自体の認知に困難性を伴う場合（本質的な認知的負荷）」への対応も試みたが，これについては直接間接問わずその効果を示す結果は得られなかった．しかしながら，受講生全員が課題を達成しており，これらの教材とカリキュラムを用いた授業実践の総体としては成功している．

ここまで，コードの書き方や文法などを習得するための入門段階における演

習を対象にその分析と改善について述べてきたが，本研究では，これに加えて，応用力を涵養するための演習過程のあり方について考察するとともに，その具体策を提示し，さらにそれを実践の中で検証している．この段階における演習では，コードの用い方や文法そのものを模倣するわけではないが，導入段階では，仕様やそれに基づくプログラムの設計など，課題を構成する内容の教授法が問題となる．

著者は，応用力を涵養する段階の学習支援のあり方として，近年，大学院や大学などの授業で取り入れられている PBL 形式の授業に着目した．本研究では，高等学校における PBL 形式のプログラミングの授業を対象とし，そこで生じている問題の対策として，問題解決プロセスを指導者とともに事例を模倣しながら学ぶためのプログラミング教材を開発した．それまでは生徒自身が設定した課題への取り組みに困難が生じていたことに対し，新たに開発したプログラミング教材を用いた授業実践では，例示したものを参照しつつ，生徒自身が創意工夫を行いながら，全員が自身の目的とするプログラムを開発させることに成功しており，実践的なプログラミング能力の涵養を目的とする授業としては，一定の効果が確認された．

Study of Computer Programming Education for Novices Focusing on Importance of Imitative Learning

Masako OKAMOTO

Abstract

Computer programming education has been widely practiced in many vocational programs in secondary and higher education to train engineers in information technology. Further, many appeals to society for the importance of programming have been made by famous people in IT fields, and some companies have started to provide people with online learning environments. Computer programming education is not limited to vocational programs; rather, it is relevant to general education. In the new Course of Study for junior high schools in Japan, computer programming has been compulsory in the subjects of Technology and Home Economics while it was optional before.

In computer programming education, a vast amount of research has been conducted, and various improvements have been proposed. However, we do not have standard guidelines for teaching, and further research and improvements are needed. This thesis focuses on computer programming education, especially on the improvement of the process of exercises.

Both exercise-style classes and lecture-style classes, i.e., classes that focus on knowledge transmission, share the goal of achieving the learning task within the class period. However, in exercise-style classes, learners must accomplish the work by themselves. They learn actively from the work, and therefore, the usability and understandability of the learning material has a more direct effect on the learners' performance. For example, learning how to write code and the grammar of a programming language through editing and executing sample codes is a typical practice in the early stages of learning. However, learners often encounter different grammatical errors and runtime errors, and they feel difficulty in fixing them. That is, the learning process is interfered not by the difficulty of the learning objectives themselves but by the skills

needed to accomplish the task. It is also observed that the learners acquire a poor understanding of the learning objectives while they accomplish the tasks. Hence, understanding the support needed for the accomplishment of learning tasks is an important matter.

The author focused on the importance of the learning process of mimicking sample programs and executing them, called "*Shakyo*-style learning", and examined the learning materials for the process exercises in computer programming in order to make them more efficient and effective.

As a first step, the author investigated a programmer training course for novices in a small company that has an interesting training method. Difficulties that were faced by novices, such as the missteps in the process of *Shakyo*-style learning, were extracted. Next, they were classified into the following three categories according to cognitive load theory, and the learning strategies corresponding to these categories were proposed:
(1) Cases where a learner cannot process the work by himself because of the extraneous or ineffective cognitive load;
(2) Cases where a learner cannot understand the work because of the cognitive difficulty of specific examples;
(3) Cases where a learner cannot understand the work because of the cognitive difficulty of learning task.

Then, the author developed learning materials taking into consideration the above findings. First, the author tried to reduce problems in category (1), and as the result, a great decrease in missteps according to the procedure with the work was observed. In addition to a decrease in missteps, the understanding of the learning task itself was also substantially improved, especially in processing the repetition in computer programming, which is a typical difficult point for novices. This result suggests the importance and effectiveness of *Shakyo*-style learning.

The second intervention that was tried involved improving the problems in category (2). The author developed learning materials using a microcomputer board focusing on "visual manifestation". This procedure was used in the undergraduate program of a university for evaluation. As a result, it was found

that the learning material enhanced learning effectiveness.

In addition to the early learning stages of computer programming in which the learners acquire the basic grammatical pattern of the programming language, the author also examined the exercise for the purpose of cultivating the computer programming ability needed to construct a whole computer program according to the learners' needs.

The author focused on PBL(Project-Based Learning), which is often used in higher education as a teaching method at this stage. In this paper, a PBL class using computer programing for problem solving in a high school is examined as a case study. In this class, students faced difficulty in constructing computer programs that they needed for their projects. The author developed a curriculum and learning materials for the students to learn the whole process of problem solving by mimicking the example case. In the course, they practiced using the developed material, and all the students successfully created programs to meet their project needs after mimicking the example cases. Therefore, the students were able to achieve positive results for the exercise process and cultivate practical abilities in computer programming.

模倣の重要性に着目した初学者向け プログラミング教育の研究

目次

図目次

表目次

第1章　はじめに

1.1　研究の背景

プログラミングは，情報技術の基本的技能であり，高校，大学，専門学校など現代の中等，高等の専門教育において，情報技術者の育成のために初学者へのプログラミング教育は数多く実践されている．さらに，IT 業界の著名人がプログラミングを身に付けることを社会に訴える活動 code.org[10] やオンラインでプログラミングを学習する環境を提供する codeacademy[9] などの企業もある．また，プログラミング教育は，専門教育に留まらず，普通教育でも取り入れられてきており，わが国の新しい学習指導要領 [35] では，中学校の技術・家庭科において従来は選択であったプログラミングは，生徒全員が学ぶ事項になった．こうしたなか，プログラミング教育は，広い学習者層に対応したものとして，その初学段階から見直すべき時期に来ているものと著者は考える．

もちろん，Pears らの調査に見られるように [1]，プログラミング教育の初学段階を対象とした研究は数多くあるし，様々な改善も試みられてはいる．しかしながら，その教授法について標準的な指針が示されるまでには至っておらず，さらなる研究と改善が必要とされているというのが現状である．

コンピュータプログラミング教育は半世紀以上にわたって実施されてきたが，とりわけ高等教育機関で FORTRAN や C 言語などの高水準言語を学習する場合については，1980 年代を跨いで，その学習環境を大きく変化させている．コンピュータの普及と技術革新によって，学習形態は，座学中心だったものから演習を大幅に取り入れた態様に変化したのである．現在では，あまり時間を気にすることなく，幾度でもプログラムをコンパイルして実行することができるが，かつてのように大型計算機を使用する環境では，コンピュータは各人が自由に使用できるものではなかったし，コンパイルを完了するまでに数時間はおろか数日を要することなど珍しくなかった．こうした事情から，日常的に演習を実施することは難しく，授業は専らテキスト教材の解説を中心に実施されることが多かったが，コンピュータプログラミングを巡る学習環境が変化した現在では，演習を中心にした授業構成が一般的である．こうしたことなどから，本研

究では，プログラミング教育に係る改善を目的として，とくにその演習過程に着目した．

1.2 研究の目的

演習形態での授業は，当該授業の期間内に学習課題を達成することを目標とするという点において，知識伝達型や座学形態の授業と相違はないが，学習者自身が作業を進め，また，その作業から能動的に学ぶという学習形態であるため，教材そのものの使いやすさだったり，分かりやすさが直接的に影響する場面が散見される．

例えば，プログラミング学習の初期段階では，サンプルプログラムを写して実行するという模倣の過程を通して文法やコードの書き方を学ぶことが広く行われているが,「プログラミングを学び始めた初学者は，間違っている箇所を自分で発見できずデバッグ（プログラムの誤りを探して修正する作業）ができないため，ここでつまずくことが多い」（[45] や [12]）と指摘されているように，ここでは，タイプミスに起因する文法エラーや実行時のエラーを多発する学習者が見られ，エラーを修正できないために学習そのものが停滞してしまう場合もある．このように，当該学習課題そのものではなく，学習のためにプログラミングの作業を遂行していく過程に問題を抱える場合が少なからず見られる．また，作業は遂行できても，これを介して学習課題を理解することにはつながらない場合も見られる．こうした点から，演習を授業として成立させるためには，教材やカリキュラムなどへの配慮など，自立的な学習の遂行やそれを介した課題の理解への支援が重要であるといえる．

これに対して，著者は，プログラミング学習のうち，サンプルプログラムを写して実行する模倣の過程をとくに「写経型学習」と位置付け，初学段階における演習過程を対象に，その困難性について綿密に調査するとともに，教材の改善に係る具体策を提示し，さらにそれを実践の中で検証していくこととした．

ただし，プログラミングの基礎を習得したとしても，それだけでは，自ら新たなプログラムを構想し，そして作成する実践的なプログラミング能力を身に付けたとはいえない．例えば，皆月は,「授業時間を長くしても実践的なプログラミングのスキルの会得は難しい」[34] としているし，和田 [56] は,「アルゴリズムや文法だけを教えるのであれば，個々のテーマごとに最適の例題や演習課題

を用意するだけで，十分に対応することができる．市販されている教材の多くは，このような形式に基づいて記述されている．このような細切れの課題対応型に期待できるのは『コードの書き方』の修得までであろう」と言及している．

この段階における学習のあり方として，著者は，近年，大学院や大学などの授業で取り入れられている PBL（Project-Based Learning，以下「PBL」) 型学習に着目し，大学院における PBL の実践例についての調査（[27] や [28]）から得られた知見のほか，藤岡ら [13] などの報告を参考に，PBL 形式のプログラミング教材を開発した．本研究では，同教材を用いて，問題解決プロセスを指導者とともに模倣しながら学べるようカリキュラムを構成し，これを PBL 型学習を取り入れている高校において実践した．

この段階における演習では，コードの用い方や文法そのものを模倣するわけではないが，導入段階では，仕様の設計やアルゴリズムの構築など，課題を構成する内容の一部については，教員側からの提示が必要とされる場合があり，ここでもまた演習における模倣のあり方が問題となる．

このように，本研究は，演習における模倣の在り方に着目して，教材やカリキュラムの開発などから授業の改善を試みたものである．

1.3　本論文の構成

本論文は，7 つの章より構成されており，第 1 章では，本研究の背景と目的について述べた．

第 2 章では，プログラミング初学者がサンプルプログラムにある命令，文法などを模倣する「写経型学習」過程を中心に据えた教育を行っている企業に注目し，この学習過程での問題点を実践状況について参与観察する調査を行い，プログラミング初学者に見られる学習者の困難性（潜在的なつまずきの要因）を抽出する．

第 3 章では，第 2 章で得られたプログラミング初学者のつまずきについて，認知的負荷理論に依拠して類型化するとともに，それぞれに対する学習方略について検討する．

第 4 章では，第 2 章 で得られた知見を踏まえ，写経型学習を進める際においてつまずかず，プログラムの記述と動作の関係から自ら学び取っていくことをを可能にするために，教材を開発し，大学での教養教育科目を対象に，2009 年

度の授業において検証を行い，2011 年度の同科目においてさらにいくつかの改善を加えた教材の有効性を検証する．

第 5 章では,「対象の認知と現象の把握」の過程に着目し，プログラムと動作の関係を視覚的に「顕在化」することに配慮してマイコンボードを活用した教材を開発し，2010 年度の大学での教養教育科目での運用を試みた結果をまとめる．

第 6 章では，高等学校における PBL 形式の授業実践を通して，実践的なプログラミング能力の涵養に係る具体的方略やそこで得られた問題について詳述する．

第 7 章では，本論文のむすびとして，これまでの考察を総括し，プログラミング演習過程の最適化に係る実践的課題の抽出とその展望を整理する．

第2章　模倣を重視した学習過程とそこでの初学者のつまずき

2.1　研究の背景

現在，コンピュータプログラミング学習において，演習は，そのカリキュラムの中心的な学習過程のひとつとして取り入れられており，まず，入門段階において文法やプログラムの書き方を学ぶ際に，サンプルプログラムを写して実行するという模倣の過程として実施される．本研究では，この演習をその他の実習や演習と弁別するために「写経型学習」過程と呼称することとする．

特に，入門段階にある学習者にとっては，他の方法を併せ用いるとしても，この段階を踏まずしてプログラミング技術を身につけることは難しいと著者は考える．実際，ほとんどの初学者を対象としたテキスト教材（例えば，[49] や [55] など）では，プログラムの概念的，理論的説明のほかに，サンプルプログラムを読むこと，実行結果の画面キャプチャ（画像）を見ること，そして最後に，実際にプログラムを打ち込んで動作させるといった作業を要求している．単にプログラムの文法や命令の機能を理解させたいだけであれば，この学習過程のうち,「概念的説明と実際のサンプルプログラムを読む」ほか,「実行結果の画面キャプチャを見る」程度で十分であり,「実際にプログラムを打ち込んで動作を確認する」必要はない．それにも関わらず，あえて命令を入力させ，動作させ，その結果を確認させるのは，プログラミング学習が書いてあるプログラムの意味を理解するだけではなく，目的とするプログラムを組み，そして実際に動作させることが出来るまでのスキルの習得を目的としており，このような演習過程に学習効果が期待できるためであると考えられる．

これまで，プログラミング学習におけるつまずき箇所のひとつとして，この過程における課題が個別に取り上げられることはあった．しかしながら，この演習過程を対象にして包括的に改善しようとする学術的な取り組みはほとんどなされてこなかったし，もちろんそうした報告も見当たらない．加えて，演習過程を重視するテキスト教材（例えば，[23]）は数多く出版されており，各著者が各々その方略の有効性やテキスト教材の全体の構成やレイアウト等の編集時

の工夫を述べてはいるものの，学術的な根拠までは示されていない．このように,「プログラムを模倣して実行する」といった入門段階の演習過程については,それがごく一般的で教える側の暗黙上の共通認識となっているため深く省みられてこなかったものと推察されるが，それゆえまた，プログラミング学習において根源的でかつ一般性の高い研究対象であると著者は考える．

2.2 研究の目的と方法

これまで，プログラミング学習を巡る研究のなかで，そのつまずきや問題点に言及したものは少なくない．しかしながら，それらは，部分的あるいは個別的な案件を抽出しようとした試みであり，演習過程全体を一連の過程として捉えてはいなかった．写経型学習過程は，テキストを見てキーボードから入力するところからはじまる動的で複雑な一連の作業から構成されている．こうした動的な作業過程を包括的に把握しようとした場合，つまずきの抽出においても,単にどの学習課題でつまずいたのかを把握するだけでなく，それに付帯する状況を全体のなかで位置づけることが必要であると著者は考える．こうしたことから，本研究では，フィールド情報学の分野で参与観察の手法がもちいられるように，この手法を用いてつまずきの抽出を行い，それらを分類することで個々のつまずきの位置づけを把握しようと考えた．フィールド情報学において，参与観察は,「フィールドにおける人々の行為や振る舞いが人工物や他の人々と複雑に絡み合いながら相互的に構成されている様子を記述しモデル化する手法としてエスノグラフィは有用」[26] とされているように，一連の動的で複雑な対象を総体として把握しようとしたときに有効な手段であるとされる．

こうしたことから，本章では，実習を中心としたカリキュラムで初学者に C 言語を学ばせているソフトウェア会社・A 社の研修を対象にして，著者が直接,参与観察の手法を用いてそこでの学習者のつまずき箇所とそれに付帯する事象を抽出することを試みた．

2.3 企業における初学者のプログラミング研修

本研究では，大学等の学校教育でのプログラミング教育をその改善対象としているが，A 社は，組み込みソフトウェアの開発などを主な業務とするソフト

ウェア企業である．同社では，自社での人材育成を目的に，これまでプログラミング経験を持たない初学者を対象として，短期集中型のプログラマ養成のための研修を実施してきている．2007 年度〜2009 年度に，京都大学との共同研究として，その研修の評価や教材の開発に取り組んだ．

同社では，プログラミング初学者に対し，1ヶ月間のプログラミング研修を実施しており，同研修の特徴として，以下のことが挙げられる．

(a) プログラムの動作や文法の意味の理解に先立って，実際に教科書 [49] の例題のコードを入力，コンパイル，実行する経験を多く積ませる指導を行っている．多くの科目を並行履修する大学，大学院の教育と異なり，プログラミングの学習に時間を使える企業内教育の利点を活かして，数多くのプログラムを入力し，実行することを義務付けているところが特徴である．この行為により，学習者は，プログラミングの作法などを反復しながら習得する．

(b) プログラミングに関する基本的な概念の学習後に，課題を与えて自ら解決させる PBL 型の学習を取り入れている．

(c) 同研修では，一斉講義は行わず自習方式を採り入れており，研修者は教材に沿って学習する．また，研修者のつまずきや問題点に関しては，研修担当者が，自らの経験を踏まえて個別に対応し，指導している．

教科書の内容をすべて終了した時点で，学習者の知識の定着，活性化のために，小規模なプログラムを要求仕様に沿って作成する最終テストを実施している．これらを通して，単にプログラミング言語の文法の獲得だけではなく，新しい機能・言語などの知識を自立的に獲得できる能力を身に付けたプログラマも短期間で育成してきている．

このように，同企業では，写経型学習を重視した初級プログラマの研修を実施しており，テキスト教材の説明部分を読んで理解する過程に加え，サンプルプログラムを改変して試す過程についても個別学習に委ねており，各学習過程において不明な点や自ら解決できない箇所についてのみ研修担当者に質問するように指示していた．

2.4 企業研修の写経型学習過程における初学者のつまずき

A 社の研修が，プログラミングの学習という点では，高校卒業以上のプログラミング初学者を対象としている点では大学の授業における受講生とで条件が

一致していること，また同社の研修が興味深い方法で一定の成果を得ていること，企業内での教育という視点から学校教育を相対化することで新しい気付きを得る可能性があることから同社の研修を調査対象に選んだ．

著者は，2008 年には，1 月から 3 月までの 3 月間にわたり，研修生 5 人を対象として参与観察を行った．なお，同研修生は，20 代から 30 代のプログラミング未経験者である．また，外部からの観察によって把握しきれないであろう研修生の思考などの内的側面については，インタビュー調査で補った．インタビュー方法は，研修の過程に沿って，行った内容などを逐次確認しながら，その過程を振り返ってもらい，各段階で研修者が感じた困難さを抽出する方法を取った．インタビュー対象者は 2 名であり，振り返りには，2 時間程度のインタビュー 5 回を要した．実施期間の制約からインタビュー対象者は 2 名に限って実施し，インタビューで得られた状況とこれまでの経験との整合性などに関しては，研修担当者等，同社の社員に併せてインタビューを行い確認した．

上述した調査の結果から，得られたつまずき箇所や問題点については，以下のとおり抽出できた．

それらは，概して「学習環境や学習者層などの条件によって生起する周辺問題」と「演習の中心課題としてのつまずき」の 2 つに大別できる．

(1) 学習環境や学習者層などの条件によって生起する周辺問題

事例１）プログラミング言語に関わる用語や記号などの音読ができないために，研修者が担当者とのコミュニケーションを円滑に行えないという問題が生じていた．

事例２）「変数」は,「文字 (列)」であるという認識をし，数値などの値を「代入」して用いることが理解できなかった．つまりプログラム以前の問題として，変数の概念を知らなかった．

(2) 演習の中心課題としてのつまずき

事例１）コンパイルする命令とプログラムを実行するための命令を混同していた．

事例２）「expected ‘ ; ’ before ‘ return ’」という「return」という命令の前

に文法エラー (syntax error）があるという表示から該当箇所を発見することができなかった.

事例３）「致命的エラー：studio.h: No such file or directory」という「#include⟨stdio.h⟩」の綴りが間違っているという文法エラー（syntax error）があるという表示から該当箇所を発見することができなかった.

事例４）条件分岐処理において，論理演算子を用いて記述する場合と入れ子で記述する場合を混同していた.

事例５）実行結果に小数点が表示されない（作成したプログラムでは，変数の型と出力の形式が異なることについてテキスト内に説明がなかった）ため，実際に入力したプログラムの内容と画面に表示される実行結果の関連付けができず，何を学んでいるのかが理解できなかった.

2.5　抽出されたつまずきや問題点への対処

抽出されたつまずきの (1) では,「学習環境や学習者層などの条件によって生起する周辺問題」を取り上げたが，例えば「事例１」の音読については，学習者が自身で演習を進めていく過程においては必ずしも理解していなければならないものではないが，本研修のように個別の演習に入る前に講師による音読を聴取する機会がない場合に生起する問題であり，これまで必ずしも十分に意識されてこなかった側面である．しかしながら，講師と学習者がコミュニケーションを取る場合には読み方を知っている必要があるため，プログラミングで使用する記号や予約語には読み方（ルビ）を表記したり，それに対する日本語訳を併記することなどによって対処すべきであると著者は考える．また,「事例２」については，高等学校までの学習内容である数学の基礎知識の不足を伴ったものであり，対象とする学習者層に応じて解説欄を設けるなどの対応が考えられる.

一方，(2) に関しては，直接的に演習に係るつまずきを列挙したものであるが，これらのつまずきについては，次章以降で認知的負荷理論の観点からさらなる分析を加えて類型化し，それぞれの類型に対応する学習方略について，いくつかの研究事例を紹介したい.

2.6　本章のまとめ

A 社での初学者を対象とするプログラミング研修は，以下のような特徴を持っており，この点は，大学など学校教育への適用が可能であると考えられる．
(1) 学習者は，教材を用いた学習を基本とし，指導者は，学習者のつまずきへの支援を個別に行う方針で実施していること
(2) 学習者は，教材にある例題をすべて実際に入力し，実行することを求めている．
(3) 知識の定着，活性化のために，教材の内容を修了した時点で要求仕様に沿った小規模なプログラムを作成することを求めている．

他方，筆者による参与観察から，上述した学習の実施において，学習者は様々なつまずきに出会っていることが判明した．本研究では，同社の実践から，例題のプログラムを模倣することが写経型学習の重要な点であると捉えるとともに，そこでのつまずきを軽減することを第３章から第５章で考えていく．また知識の定着，活性化の段階，つまり，応用力の涵養する段階では，同社の実践はある種の PBL として捉えられるが，その学習の方略としては十分に練られたものとは言い難い点もある．そこで，PBL 型の授業に取り組んでいる高等学校の事例に着目し，問題解決過程全体を例示，模倣させることで改善することを第６章で試みる．

第3章　認知的負荷理論を用いた初学者のつまずきの類型化

第2章では,「写経型学習」におけるプログラミング初学者のつまずきや問題点を抽出し,

1. 学習環境や学習者層などの条件によって生起する周辺問題
2. 演習の中心課題としてのつまずき

の2つに大別した.

本章では,「演習の中心課題としてのつまずき」をさらに詳しく考察することとし，学習者が作業を進め，そしてそこから学ぶ中で，何が障害になっているかを明確にするために，認知的負荷理論を用いて学習者に掛かる負荷を基準に細かく類型化するとともに，その類型に即して学習を促進するための方略について検討する．こうした類型化は，これまで，記載的ではあるものの，個々に報告されてきたつまずきをひとつの体系として示すことを目的とする．こうした試みは，実際の授業において生じる問題を適切に把握し，そして，その原因となる可能性がある要素をひとつひとつ解決していこうとしたときに有用であると著者は考える.

3.1　認知的負荷理論

「認知的負荷理論（cognitive load theory）」は，Sweller により提唱されたもので,「学習者に対する認知負荷に注目し，学習プロセスを分析する」というものである（[53] や [48]）．つまり，新たな知識や技能をより効果的に学習できるように，その学習がもたらす認知負荷の在り方を考えるものである．学習者が,何かを学ぶ際，要求された課題と自らの先行知識にあわせて，作業記憶の容量を適切に分割して利用しなくてはならない．このような前提をもとに，その作業記憶にかかる認知的な負荷を教材や学習活動によって軽減することができないか，教授デザインや学習方略として検討する必要がある．また,「認知負荷」とは,「学習者がある課題を処理する際にどれほどの認知処理容量を割り当てる必要があるのかを指す概念である」[53].

認知的負荷理論では，学習者が認識する認知的負荷には次の 3 種類があると考えられている．

(1) 本質的な認知的負荷（intrinsic cognitive load）：

学習課題そのものの複雑さとその課題に対する学習者の熟達化の程度によって決まるものだとされる．

(2) 非本質的な認知的負荷（extraneous or ineffective cognitive load）：

学習課題それ自体の理解とは無関係で，学習にとって余計な負荷となるもの．

(3) 適切な認知的負荷（germane or effective cognitive load）：

非本質的な認知的負荷とは対照的に，当該の学習に対して有益な貢献をする活動に起因する負荷のこと．

認知的負荷理論を用いた研究として，Mayer らのマルチメディアを利用した学習法の提案を挙げることができる．Mayer らは，認知的負荷理論に基づき，「モーターの仕組みを学ぶ際の認知的負荷を軽減させるマルチメディアの使用法の提案」をしている [33]．ここでは，パーソナルコンピュータ上に図とナレーションを同時に提示し，自己説明を促すと学習者が学習の要点に注目するようになり，認知負荷を軽減する効果があることが示されている．

3.2 「作業の自立性」に係るつまずき

プログラミング学習では，プログラムを構成する要素となる概念が順番に紹介される．学習者は，例題の実行や改変を行いながら概念の理解を進めていく．ここでは，順次紹介されていく概念を学習課題と捉える．

第 2 章では，学習課題の学習において生じる文法エラーの修正に係るつまずきについて述べたが，プログラミングでは，文法エラーの発生は珍しいことではない．しかしながら，初学者にとって今まで経験したことのない過程であるため，決して容易な作業とはいえない．例えば，コンパイル（作成したプログラムをコンピュータが実行可能な形式に変換する）操作の際，ほとんどのプログラミング言語の開発環境では，タイプミスに起因するメッセージは専門用語を含む英文で表示されるが，これを手掛かりにエラーを発見するという作業そのものの方略を知らなければ対処できない．

また，うまく実行出来たとしても,「プログラムの実行結果が想定していた結果と異なっていた」あるいは「目的とする実行結果が得られていない」というこ

ともある．このように，実行はできたが所期の動作が得られなかったというような場合には，目的に対して間違った記述でも，文法上の誤記がないため，当然ながらエラーメッセージが表示されない．この場合，学習者が厳密に結果を再現しなければならないという意識をもっていなければ，誤ったことにすら気付かないことがあるだけでなく，気付いた場合でも，修正を必要とする箇所を探す手掛かりが示されていないので，文法上のエラー修正に劣らず困難性を伴う．

上述したような文法エラーや実行時のエラー箇所を自分で発見できずデバッグ（プログラムの誤りを探して修正する作業）ができないなどのつまずきについては，[12] そして [45] などによって指摘されている．このように当該学習課題そのものではなく，それを学ぶために必要とされる先行知識の不足を原因とするつまずきは，学習一般において広く見られるもので，認知的負荷理論においては,「非本質的な認知的負荷」のひとつとして類型化されている．とりわけ，プログラミングの学習では，これらの内容が明示的に教材などで取り上げられることが少なく，授業などの教育現場で各々対応されているのが現状である．

3.3 「作業を介した理解」に係るつまずき

写経型学習過程は，提示されたプログラムの実行それ自体ではなく，その過程を通してプログラムの基本構造を理解することが主たる目的である．したがって，滞りなくコンパイルを済ませて目的とする実行結果を確認できたとしても，それで学習目標を達成したことにはならない．具体例による学習は，具体例の実行を通してその背後にある基本構造を理解するとともに，同様の具体例を自ら構成できるようにすることを求めるもので認知科学的視点から見ると,「宣言的知識を利用可能な知識へと変換することを促進する」[46] とされている．写経型学習過程とは，すなわち，宣言的知識を定着させ，知識スキーマを構築していく過程に該当するわけである．しかしながら，この過程を通して学習課題の基本的な要求事項それ自体の理解に至らない場合も少なくない．例えば，著者は，写経型学習過程を進める中で「何が何だかわからぬまま，指示されたとおりのことをこなしていた」[41] とするアンケート結果を得ている．また，その要因等については，場合によって異なるためここでは触れないが，写経型学習をともなう実践報告において,「順次処理（プログラムの命令を上から順番に実行すること）を理解することでさえも初学者には難しい場合がある」[40] ほ

か,「for 文の入れ子的な利用も初学者が典型的に困難さを抱えるものである」[2] などの指摘がある．これらは，不適切な教材を使用することなどによる「非本質的な認知的負荷」に起因する場合や，学習課題自体の複雑性などの「本質的な認知的負荷（instrinsic cognitive load）」に起因する場合が想定される．

ここで，前章で「写経型学習」におけるプログラミング初学者のつまずきや問題点について提示した内容を一部改変，および新たに認知科学的視点を追記して表 3.1 のとおり示したい．

表 3.1 では，写経型学習に係るつまずきを

1. 写経型学習を遂行する上で自立的に作業することができない「作業の自立性に係るつまずき」
2. 写経型学習の過程から目的とする内容を学び取ることができない「作業を介した理解に係るつまずき」

の２つに類型化した．

さらに，ここでは，先行知識の学習に及ぼす負荷を認知的負荷理論を用いて

1. 本質的な認知的負荷（intrinsic cognitive load）を伴う学習過程によるつまずき
2. 非本質的な認知的負荷（extraneours or ineffective cognitive load）を伴う学習過程によるつまずき

の２つに区別した．前者は，学習課題を構成する情報の間に見られる関係性の複雑さと，その課題に対する学習者の熟達化の程度によって決まるものだとされる．後者は，実際に学習内容を理解していくためには本質的には関係ない活動を強いることによって学習者の中に生じるものであり，学習課題それ自体の理解とは無関係で，学習にとって余計な負荷だといえる．

3.4 つまずきの各類型に対応する学習方略

3.3 では，著者がプログラミング教育の場で調査したつまずきを類型化して示したが，本節では，これら３つの類型に係るつまずきの原因や対応する学習方略について指摘あるいは提案されている研究をそれぞれ紹介する．

表 3.1: プログラミングの「写経型学習」過程におけるつまずきの類型化

	本質的な認知的負荷を伴う 学習過程によるつまずき	非本質的な認知的負荷を伴う 学習過程によるつまずき
「作業の自立性」に 係るつまずき		【潜在的なつまずきの要因】 学習支援環境の不備など 【事例】 実行するための命令「a.exe」がわからない ⇒ 「実行結果を確認できない」
		【潜在的なつまずきの要因】 学習支援環境の不備など 【事例】 「expected ‘;’ before ‘ return’」という「return」という命令の前に文法エラー(syntax error)があるという表示から該当箇所を発見することができない ⇒ 「エラー発生時に修正が必要な箇所を発見することができないのでエラーの修正ができず実行結果を確認できない」
「作業を介した理解」に 係るつまずき	【潜在的なつまずきの要因】 複雑性あるいは類似の概念を学んだことがないという学習者にとっての概念の新規性 【事例】 条件分岐処理において、論理演算子を用いて記述する場合と入れ子で記述する場合を混同している	【潜在的なつまずきの要因】 学習支援環境の不備など 【事例】 実行結果に小数点が表示されない(作成したプログラムでは、変数の型と出力の形式が異なることについてテキスト内に説明がなかった) ⇒ 「実際に入力したプログラムの内容と画面に表示される実行結果の関連付けができず、何を学んでいるのか理解できない」

3.4.1 「作業の自立性」に係るつまずきと対応する学習方略

学習者がプログラミングを学ぶ際，コンパイルの手順やエラーの解決方法については，先行知識として持っているに越したことはないが，必要に応じて学習者に提供したり，あるいは，学習者が必要に応じて参照できるテキスト教材を用意すれば良いわけで，教授者にとって対応はさほど難しくない．

著者は，こうしたつまずきの排除を主な目的とした C 言語の初学者向けプログラミング教材を開発して実践で使用したところ，手順に係るつまずきが大幅に減少し，それに伴ってプログラムの概念そのものについても理解が大きく促進されたという結果を得ている（第 4 章参照）．同教材では,「コンパイル手順やエラー発見の方法に関する手順」に関してテキスト本体とは別の冊子にまとめ，学習者が逐次参照しながら作業を行えるようにしたほか，同冊子に掲載するエラーの事例を増加させるなどしているが，こうした軽微な改善で顕著な学習効果を得ている．

これは，教材の使いづらさや説明不足といった「非本質的な認知的負荷」が要因となって生じる「作業の自立性」に係るつまずきを排除あるいは軽減することによって改善された事例である．

3.4.2 「作業を介した理解」に係るつまずきと対応する学習方略

「作業を介した理解」については，大別して 2 つのつまずき要素が内在していると著者は考える．ひとつは，具体例の認知の困難性に伴うもの（非本質的な認知的負荷）であり，もうひとつは，当該学習課題そのものの認知の困難性に伴うつまずき（本質的な認知的負荷）である．本章では，この類型に従って対応する学習方略を紹介する．

作業を介した理解に係わるつまずきにおいて具体例の認知に困難性を伴う場合（非本質的な認知的負荷）

写経型学習では，記述（命令）が示す動作を確認する過程においてその学習目的を達成しようとするならば，動作は記述を反映したものとして認識されるよう明示されていることが望ましい．しかしながら，学習に使用されているサンプルプログラムの中には，当該命令の実行が動作として顕在化していないために，動作を確認しただけでは当該命令との対応関係を認識出来ないものが散見される．この場合，学習者に対し，潜在的な動作について推察する思考過程を要求することになる．推察や判別については，それを行うか否かを含め，学習者個人の習慣や類似する先行知識の有無，そして意思などを含めた総合的能力に依存するものであるし，加えて，推察や判別それ自体が認知的負荷となって学習者の認知資源（cognitive resource）を余計に消費することになる．具体例を工夫することによってそれが直観的に理解できる対象に改善できるのであれば，推察や判別の要素は，学習者にとって非本質的な認知的負荷だったといえる．当然ながら，認知資源の多くを学習課題の本質的部分を理解するために充てるべきであって，推察や判別の要素についてはできるだけ排除することが望ましい．

こうした視点から，著者は，動作を認知する際の推察や判別の要素をできるだけ排除し，第 5 章で後述する「視覚的顕在化」という概念にしたがって，非本質的な認知的負荷の排除を試みた教材を開発し，同教材を用いた実践でその効果を確認している．

また，数学教育の分野では，Kaminski ら [20] が,「具体的事例から抽象的要素を抽出することが難しく，事例のもつ物語性が概念の抽出を阻害している」と

考察しているが，サンプルプログラムが具体例の学習であるという側面を捉えるならば，写経型学習にも同様の困難性が想定される．

作業を介した理解に係るつまずきにおいて当該学習課題それ自体の認知に困難性を伴う場合（本質的な認知的負荷）

本節では，当該学習課題そのものの認知の困難性に伴うつまずきについての報告事例あるいはそれに対応した学習方略について紹介する．河内谷 [21] は，特に for 文の学習について,「類似概念をこれまでに目にしたことがない」こと,「多くのデータから判断を下す，かなり複雑な問題」であり,「認知的負荷が大きい」ことがその主たる要因ではないかと考察している．さらに，同文献では，類似概念が既習でない場合について，当該概念を構成している複数の要素を分解して提示し，個々の要素に類似の概念を照らし合わせて説明することで理解に導くことができるのではないかと提案している．

また，長谷川らは，プログラムの動作を条件によって切り替えたり，同じ動作を繰り返して行うなどの「制御構造」について，そのイメージを自由に絵で表現する方法で，学習者のイメージ（二次元抽象図）を調査した結果，プログラミングを習得するには，プログラムが実行された際に処理の流れの明確なイメージを持っている，つまり，実行時の動きを抽象化して捉えたイメージを持っている学習者ほど，プログラミングの理解度が高いことを示している [17]．これに続いて長谷川らは，初学者を対象とした授業のなかで，制御構造を表す図を提示し（図 3.1 参照），処理の流れのイメージを形成されることによってプログラミングの理解を支援することができ，学習効果が得られることを明らかにしている [18].

さらに，著者（詳細は第 5 章参照）は，制御構造の条件分岐処理を学習する際に実物のスイッチを組み込んだマイコンボードを使用しているが，これは，実物のスイッチが条件分岐の概念を容易にイメージさせるとの考えから採用したもので，類似のイメージを提示することによって本質的な認知的負荷を軽減させようという試みである．

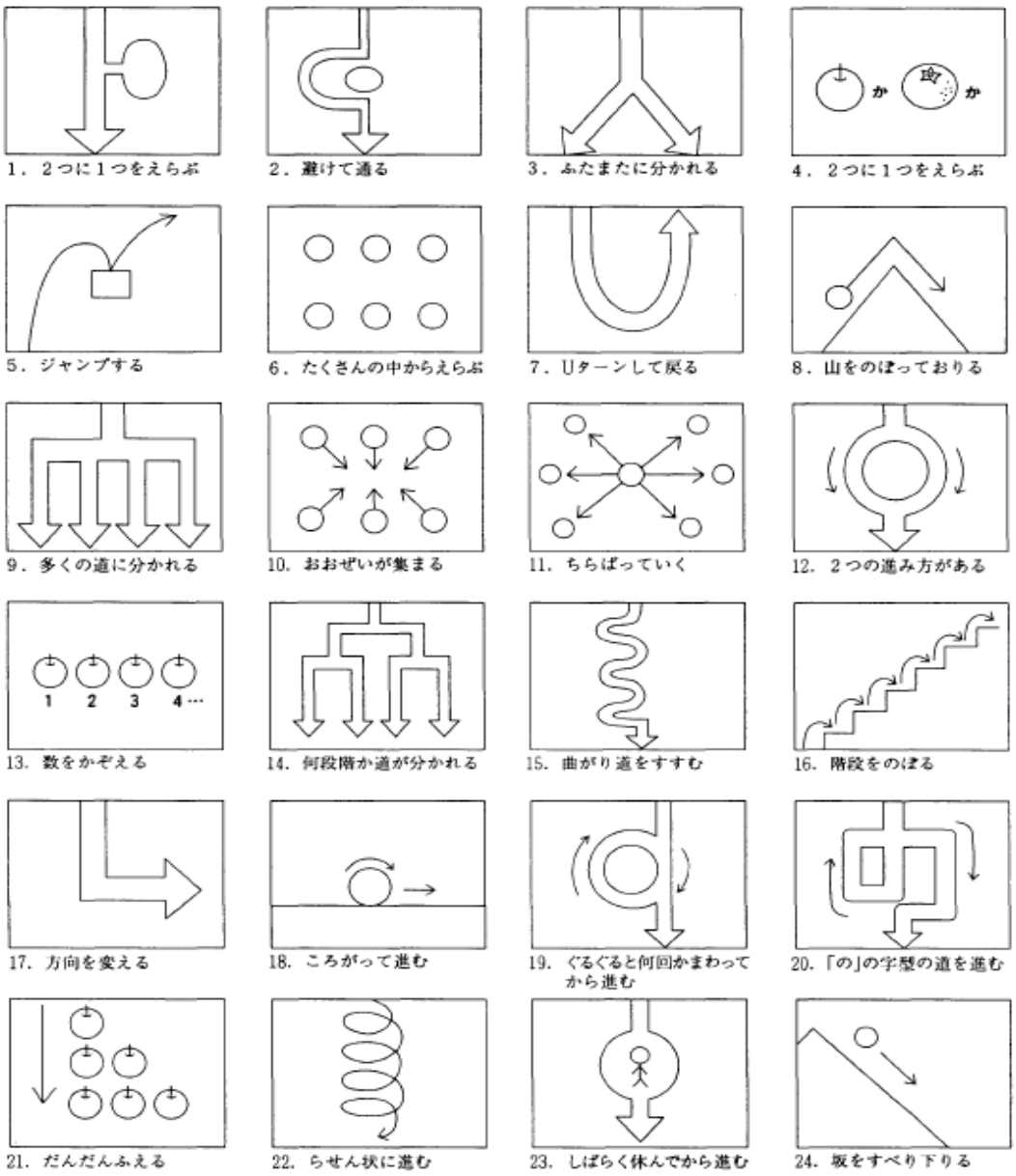

図 3.1: イメージ図形の選択肢（出典 [18]）

3.5 本章のまとめ

本稿では，プログラミング学習の初学者が「写経型学習」過程において見られるつまずき事例を

1. 写経型学習を遂行する上で自立的に作業することができない「作業の自立性に係るつまずき」
2. 写経型学習の過程から目的とする内容を学び取ることができない「作業

を介した理解に係るつまずき」

の２つに類型化し，さらに，認知的負荷理論を用いて，

1. 本質的な認知的負荷（intrinsic cognitive load）を伴う学習過程によるつまずき
2. 非本質的な認知的負荷（extraneours or ineffective cognitive load）を伴う学習過程によるつまずき

に分けて対応する学習方略を紹介および提示している．

とりわけ，非本質的な認知的負荷については，表 3.1 に示したように教材の使いづらさや説明不足が要因となって生じる場合が散見されることから，手順などについては記載のとおりに実行するだけでよいように提示したり，具体例を適切に選択するなどの工夫によってその排除を試みることを提案したい．

第4章　「作業の自立性」に係るつまずきに配慮したテキスト教材の開発

4.1　研究の背景

第2章で述べたように,「写経型学習」過程は，プログラミング初学者が一般的に取り組む過程である．ここでは，文法事項などについて解説を読んだり，説明を受けた後に実施されるのが一般的で，文法などの知識を確認したり，さらに進んで，個々の知識を有機的に結び付け，利用可能な知識として再獲得する重要な学習機会であると著者は考える．この演習過程では，入門段階にある学習者が自立的に作業を遂行していくなかで，その作業を介して学習課題となる概念を理解していく．しかしながら，第2章で述べたように，学習者が作業そのものを自立的に遂行することができず，つまずいてしまうことが明らかとなっている．

4.2　研究の目的

本章では，教材の使いづらさや説明不足といった「非本質的な認知的負荷」が要因となって生じる「作業の自立性」に係るつまずきを排除あるいは軽減することによって，プログラミング初学者を支援することを試みるため，第2章で述べた企業研修の調査やその分析から得られた知見をもとに,「写経型学習教材」を開発し，2009年度の京都大学の全学共通科目「情報科学演習」において検証を行い，2011年度の同科目においてさらにいくつかの改善を加えて再度開発した教材の有効性を検証した．

4.3　関連研究

プログラミングの初学者は，学習の初期段階でつまずくことが多い．とりわけ学習の初期段階では，プログラムのタイプミスに起因する文法エラーが多発する．ここでは，コンパイル（作成したプログラムをコンピュータが実行可能

な形式に変換する）操作の際に，タイプミスを示すメッセージが専門用語を含む英語で表示されるため，初学者にとっては，用語，内容ともに初見であり，その意味を理解するのが困難である．さらに次の段階として，自分が作成したプログラムの実行結果が予想していた結果と異なるということもある．ここでは，先述した文法エラーとは異なりエラーメッセージが表示されないので，学習者が間違っている箇所を見つけられないことが多い．

このような文法エラーや実行時のエラー箇所を自分で発見できずデバッグ（プログラムの誤りを探して修正する作業）ができないなどのつまずきについては江木 [12]，そして太田 [45] などによって指摘されている．また,「プログラミングの基本である順次処理（プログラムの命令を上から順番に実行すること）を理解することも，初学者には難しい場合がある」[29],「条件式をどのようにつくればよいのかわからずに，ここで理解できなくなる学生も多い」[15]，そして「for 文の入れ子的な利用も初学者が典型的に困難さを抱えるものである」[2] などの指摘がある．また，河内谷 [21] は，for 文の学習は認知的負荷が大きいとして,「類似概念をこれまでに目にしたことがない」こと,「多くのデータから判断を下す，かなり複雑な問題」であることがその主たる要因ではないかと考察している．

4.4 2009 年度の実践

第 2 章で述べたように，プログラミング初学者を対象とした企業研修の参与観察およびインタビュー調査から抽出されたつまずきのうち，特に写経型学習に係るものについては，以下の 2 つに類型化することができた．

1. 写経型学習を遂行する上で自立的に作業することができない「作業の自立性に係るつまずき」
2. 写経型学習の過程から目的とする内容を学び取ることができない「作業を介した理解に係るつまずき」

「写経型学習」は，学習者が自ら入力し実行する「作業遂行の自立性」と，結果を確認し，プログラムの記述と結果の関係から自ら学び取っていく「作業を介した理解」の 2 つの側面に特徴がある．そのため，これら 2 つの問題点を改善することによって，作業遂行におけるつまずきをなくすこと，また，実行結

果を見て，学習者が命令と動作の因果関係について理解できるようにすることが学習を効果的に進めるための必要条件であると考えられる．

作業を介した理解を促進するためには，具体的作業から獲得すべき抽象概念や動作のイメージなどを容易に掴み取れるようにする必要があり，具体的作業における不要な認知的負荷の軽減が求められる．そのためには，具体的作業そのものを，抽象概念の抽出やイメージの形成に適したものに改善する方法も考えられるし，事前に習得すべき抽象概念などを示しておいて，後の具体的作業でそれを実感させるといった方法も考えられる．これらの方略のうち，前者については，扇風機と類似の形状を持ち，またその機能を模倣できるマイコンボードを用いて直観的に理解できるよう工夫した組み込み系C言語プログラミング教材を開発し，京都大学の全学共通科目「情報科学演習」において 2010 年度の前期に実践と検証を試みた際，それぞれの命令に対応する個別の動作に注意を向けることができたという結果が得られており，このような認知的効果が示唆されるなど一定の効果が確認されている（第 5 章参照）．しかしながら，第 5 章で述べる実践で示した方法は，特殊な装置を用意することとそのための特殊なプログラミング環境を用いる必要があり，一般的なプログラミング教育課程で用いようとすると，教授者側の負担も相応に大きくなるものとも考えられる．つまり，同実践で使用した方略には汎用性といった側面において問題があり，より広く用いることが出来る方略についても実践的に検証していくべきであると著者は考えた．

以上を踏まえて，本章においては，特別な学習用の機材等を用意せずに，従来のテキスト教材の利用を前提とした汎用性の高い改善方法について提示したいと考え，後者の方略を採用して教材の開発と実践を試みることとした．

4.4.1 写経型学習教材の開発

プログラミング初学者を対象としたテキスト教材では，サンプルとして示したプログラムが正しく動くことが前提となっており，エラーが起こったときの対処まで記述しているものは少ない．また，プログラミング言語の紹介に重点を置いたテキスト教材では，文法の解説に必要な部分だけを例示するような記述がしばしば見受けられるが，プログラムの動作を確認するためにはプログラム全体を提示する必要がある．このため，初学者がテキスト教材を用いて学習

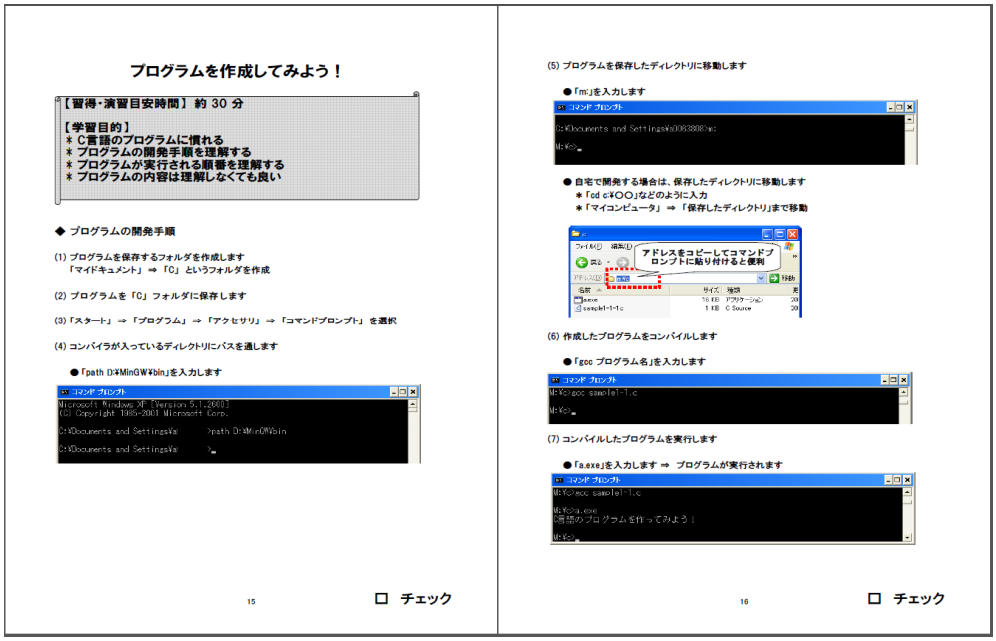

図 4.1: 手順書（2009 年度版）

した際に，一か所でもつまずいてしまうとその時点で学習を諦めてしまうことやエラー対応などに時間を費やしてしまう場合がある．

そのため，本章では，第 2 章で述べたプログラミング学習における学習者のつまずきを踏まえ，以下の 2 つの方針に基づいて教材を作成し（表 4.1 参照），同テキスト教材を 2009 年度の後期に京都大学で開講された「情報科学演習」で使用し，検証を行った．テキスト教材の開発においては，

- 自立的な作業遂行のために「コンパイルの手順やエラー発見の方法についてその手順を掲載すること」（図 4.1 参照）
- 作業を介した理解のために「学習の目的，学習の仕方など，学ぶ内容とともに学び方を明示すること」

という 2 つの方略によって問題解決を図った [24]．

なお，本章では，プログラミング言語の文法面の網羅的な紹介ではなく，学習者が実際に手を動かしてプログラミングを実践することによって理解を促すことに重点を置いており，そのような学習の際に必要とされる改善を試みている．

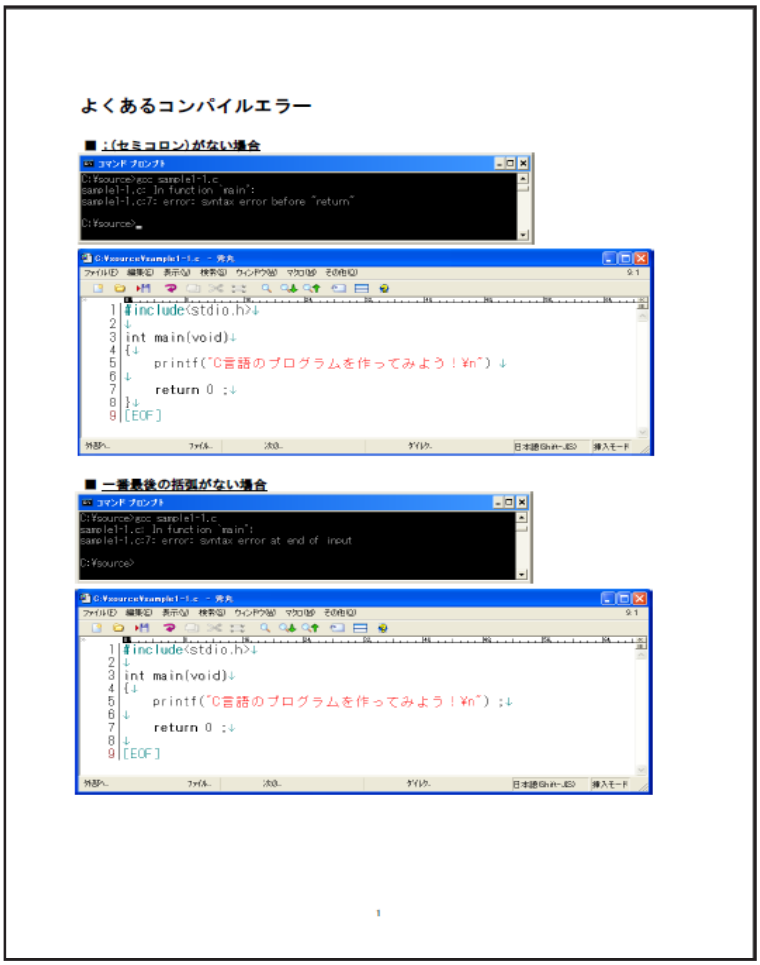

図 4.2: エラーメッセージの実例（2009 年度版）

(1) 自立的な作業遂行のための支援：

コンパイル時のつまずきを軽減するため，エラーの原因となる

「見えない文字（空白）の使用」

「大文字と小文字の区別」

「半角全角の区別」

「カッコのとじ忘れ」

などについて，本文中に注意書きを添えたほか，エラーメッセージの実例を別紙で 2 例紹介した（図 4.2 参照）.

本章では，初学者が自立的に作業を遂行することを考慮して，サンプルプログラムは「完全に動作するプログラム」と「その実行結果」を掲載する方針を取った．そして，サンプルプログラムを対象に，プログラムの各行に対応して逐

次解説を加えたほか，プログラムの構成要素の紹介も併せて行った．なお，前述の方針から，プログラミング言語の文法の詳細は他のテキスト教材等を参照させるものとし，実践的に利用するポイントに焦点を当てて解説した．

(2) 作業を介した理解のための支援：

本テキスト教材では，各章のはじめに「習得・演習目安時間」を提示するとともに，その章では何を理解しなければならないのかを「学習の目的」として，2 から 3 項目ずつ明示した．例えば，第 6 章の「値を保存する方法（変数）」では，

「値を格納する箱である『変数』について理解する」

「変数に値を格納する方法（代入）を理解する」

としている．このように，各章に「学習の目的」を説明しただけでなく，これとは別に,「学習の仕方を理解する」という章を設け，本テキスト教材を使用してプログラミングを学ぶための手順を示した．

また，プログラミングの学習が進んでいくにつれ，それまでに学習した既出項目を活用する機会が多くなり，学習者は，新たに学ぶ項目とそれまでに学習した項目の両方でつまずくことが懸念される．そのため，学習者がなるべく失敗しないように学習ステップを細かく設定し（スモールステップの原理），1 つのサンプルプログラムから新たに学ぶ項目は 2 つ以内に絞って提示した [38]．さらに，本テキスト教材では，初学者にとって難しいとされる繰り返し命令 for 文については,「for 文ドリル」という章を設け，それまでに学習した内容を組み合わせて活用した例題を数多く提示した．ここでは，プログラムの構造の理解を促すため，サンプルプログラムの内容を少しずつ改変させた例題を用意した．

4.4.2 開発した教材を用いた授業実践の対象と方法

対象とした授業

本研究の対象とした授業は，2009 年度の後期に京都大学で実施された全学共通教育科目（全学部の１回生から４回生まで受講可）「情報科学演習」である．本科目は，教養教育におけるコンピュータリテラシ教育という位置付けで週１回 90 分の選択科目として実施したもので，全体の構成は，Word による文書整形，PowerPoint と Excel の基本，HTML，CSS の基本に加え，C 言語によるプ

表 4.1: 開発したテキスト教材の目次

目次	例題数
学習の仕方を理解する	–
プログラミングに使用する文字と用語	–
C言語プログラムの開発手順	–
プログラムを作成してみよう	8
画面に表示する方法	5
データを入力する方法	2
値を保存する方法（変数）	8
処理の流れとフローチャートを理解する	–
条件によって処理を分ける方法（if 文）	6
変数を一括管理する方法（配列）と繰り返し処理（for 文）	11
配列の便利な使い方	2
for 文ドリル	74
繰り返し処理（while 文）	6
ポインタ	4
関数	7
複数のデータ型を１つにまとめる方法（構造体）	5

ログラミングに後半５回の授業を割り当てている．5 回に渡るプログラミングの授業は，プログラムの論理構造の理解を目標としており，プログラミング体験を通して「コンピュータの動作」に関する総合的理解を促すことを目的としている．このような科目構成から専門科目としてプログラミングのみを学ぶ授業に比べ，限られた時間内で効果的な指導を行うことが必要であった．

授業デザインと授業形態

対象とした科目の中でプログラミングを学ぶ機会は，先述のように１授業 90 分で計 5 回しかなく，学習する時間が限られていることから，学習内容は，プログラミングの基本的な要素である変数と制御構造を中心とし，関数や構造体は取り上げなかった（表 4.2 参照）．本授業では，教材の設計意図を踏まえ，講義と写経型学習教材による個別学習を取り入れた授業形態を採用し，講義では

表 4.2: 授業内容

授業内容	2009 年度の内容	2011 年度の内容
1 回目	C 言語プログラムの開発手順 プログラムを作成してみよう	C 言語プログラムの開発手順 プログラムを作成してみよう
2 回目	画面に表示する方法 データを入力する方法	画面に表示する方法 データを入力する方法 値を保存する方法（変数）
3 回目	値を保存する方法（変数）	条件によって処理を分ける方法 （if 文）
4 回目	条件によって処理を分ける方法 （if 文） 変数を一括管理する方法（配列） と繰り返し処理（for 文）	変数を一括管理する方法（配列） と繰り返し処理（for 文） 配列の便利な使い方
5 回目	配列の便利な使い方 まとめ	まとめ

短時間にポイントを絞って解説を行った．また，受講生は，自立的に学習できるよう開発した写経型学習教材に従って，学生間で学習の進度を合わせることなく，学生各人の進度に応じて学習を進めた．受講生がつまずいた場合は，教員やティーチングアシスタント（1 人）がサポートに入り，直接対応した．

なお，演習には，京都大学の教育用コンピュータ端末を用い，C 言語の処理系には，インストールが比較的容易で学生が自宅で作業する環境の構築が容易であることや日本語でのエラー表示機能があることなどから MinGW（Windows 版）を，プログラムの編集には汎用のテキストエディタである TeraPad を用いた．

受講生の構成

本実践での受講生は 15 人で，その学部別の内訳は，総合人間学部（6），法学部（3），経済学部（3），文学部（2），理学部（1）であった．この授業は，プログラミング初学者を想定したものであるが，経験者の履修も可能であり，本実践では，JAVA など他のプログラミング言語の既習者が 2 人であった．

検証方法

開発したテキスト教材を検証するため，受講生から教員およびティーチングアシスタントへの質問とその際の参与観察の結果における「つまずき」どころとその件数，内容について調べた．さらに，受講生のプログラミングに関する理解度を測るため，各授業終了時に「確認テスト」を，また，第５回目の授業の最後に「最終テスト」を筆記式で行った．確認テストでは，受講生の進捗状況に応じて，順次処理，条件分岐処理（if 文），繰り返し処理（for 文）に関する選択式（３択）の問題を３問，プログラムの作成問題を１ないし２問をそれぞれ課題とした．また，第５回目の授業で実施した最終テストでは，

1. 文法エラーの間違い探し
2. 変数・画面への出力（printf 文）・キーボードからの入力（scanf 文）に関するプログラムの作成問題
3. if 文に関するプログラムの作成問題
4. for 文と配列に関するプログラムの作成問題

をそれぞれ課題とした．

4.4.3 つまずき多発箇所と学習への影響

本授業では，全授業回を通して，受講者 15 人のうち既習者 2 人を除く初学者 13 人全員から延べ 63 件の質問が見られた．このうち 44 件は,「コンパイルエラー表示の意味がわからない」,「エラーの原因がわからない」など，自ら対処できないために教員に質問をしたものであり，本稿では，これらをつまずきどころとして取り扱った．これらのつまずきについて

「コンパイル手順」

「エラー対応」

「基本操作」

の３つに類型化したものを表 4.3 に示す．この類型のなかで「コンパイル手順」および「エラーの対応」と実行までの手続きに係るつまずきが計 39 件と，とりわけ多くみられた．こうした「つまずき」が多発したことなどの理由により受講生のなかには，サンプルプログラムを１回ずつ試すだけで自ら一部改変して試すことができなかった者が見られたほか，少なくとも３人はサンプルプログラムを入力しただけで，実際に動作させるまでに至らなかった．

表 4.3: 2009 年度および 2011 年度授業実践におけるつまずき件数と事例

つまずきの類型	2009年度		2011年度	
	つまずき件数	つまずき事例	つまずき件数	つまずき事例
コンパイル手順	12	コンパイルしたが前回のプログラムが実行された(コンパイルせずに実行コマンドのみ入力していた)など	1	手順書に従って進めたがコンパイルできなかった(原因不明)など
エラー対応	26	エラー表示が出たが意味が分かないため対処できなかったなど	3	不正な文字の使用を意味するエラー表示が出た際，エラー表示の意味は分かったが，エラー箇所が見つけられなかったなど
基本操作	6	・CapsLockの解除方法がわからない ・全角と半角の切り替え方がわからないなど	3	・NumLockの解除方法がわからない ・コマンドプロンプトのウィンドウを閉じるべきかそのままで実行すべきかわからないなど

4.5 2011 年度の実践

4.5.1 「つまずき」の要因に関する分析とテキスト教材の改良

2009 年度の実践では,「繰り返し処理の for 文」の理解を計る確認テストおよび最終テストにおいて，それぞれ，7 人および 11 人と初学者の過半数が誤答している（表 4.4 参照）．なお，ここでは，各問題の回答に 1 問でも誤りがあった者を「誤答者」としている．この結果は，開発した教材が，当該概念を理解するために十分でなかったことを示していることから，改善の必要があるものと考えられる．また，表 4.3 で示すように,「コンパイル手順」や「エラー対応」といった実行までの手順に係るつまずきについては特に注視して関係箇所を見直す必要があるという結果が得られた.「繰り返し処理の for 文」の両テスト結果において過半数の誤答が見られたことの原因が「実行までの手順に係るつまずき」を軽減するための工夫が足りないことであるという結論を導くことはできないが，少なくとも「写経型学習」を行う上で，実行に係る作業を滞りなく最後まで行うことは最低限必要な作業であり，学習のための必要条件を欠いていたともいえる．これらの必要条件を満たしたうえで，さらにテスト結果に良い変化が見られないのであれば,「実行までの手順に係るつまずき」以外の要素,ここでは,「作業を介した理解」の過程にも改善の要素があると考えねばならない．このため著者はまず,「実行までの手順に係るつまずき」を軽減させるための改善を進めることとした.

表 4.4: 2009 年度および 2011 年度における確認テストと最終テストの誤答者数

設問の内容		2009 年度の誤答者数	2011 年度の誤答者数
順次処理	確認テスト	0	0
	最終テスト	0	0
条件分岐処理の if 文	確認テスト	0	0
	最終テスト	3	2
繰り返し処理の for 文	確認テスト	7	0
	最終テスト	11	2

「実行までの手順に係るつまずき」を軽減させるための改善

2009 年度の実践では,「手順通りやってもコンパイルできないがどうしたらよいか」,「コンパイルエラー表示の見方（意味）がわからない」といった質問内容が多く見られたほか,
(1) コンパイルの方法が分からないと述べる受講生の一部は実際にはテキスト教材を参照せずに作業を進めていた

(2) テキスト教材を参照しながらコンパイル作業を行っていた受講生の中にも手順を進める上で混乱が生じていた．なかでも，コンパイルが毎回必要な作業であると認識しておらず，2 回目以降のプログラムではコマンドのみ入力している受講生がいたなど，作業の開始点に関して混乱が見られた．

(3) 本文中で，いくつかの典型的なエラーの原因となる
「間違いやすい文字」
「見えない文字」
「カッコのとじ忘れ」
などについて注意を促す記載があり，これらに加え別紙でエラーメッセージの実例とその解説も 2 例紹介していたが，これだけでは実際のエラーに対処できていなかった,
などの観察結果が得られていた．そこで，以下の 3 つの方針に従ってテキスト教材の改善を行い，一連の手順とエラーへの対応についてまとめた「手順書」を

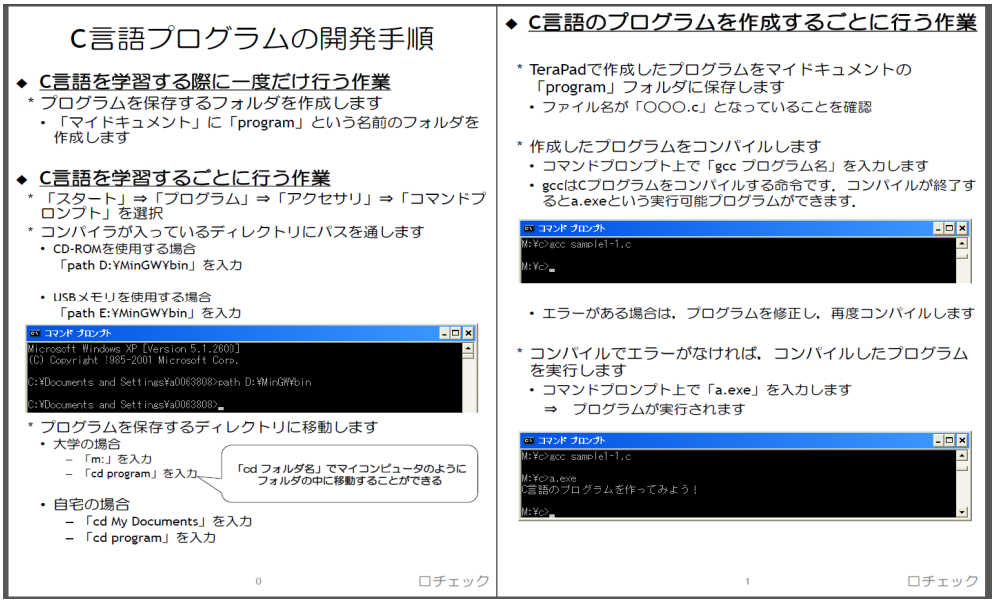

図 4.3: 手順書（2011 年度版）

別冊子として作成した．

(1) コンパイルの手順を学習の初期段階にすでに理解したはずの項目としてではなく，常に傍らにおいて参照すべきものとして認識させるため，手順書を別冊子にして受講生に配布する．

(2) コンパイルの手順を，

「C 言語を学習する際に一度だけ行う作業」

「毎回の授業開始時など C 言語を学習するごとに行う作業」

「C 言語のプログラムを作成・修正するごとに行う作業」

として，作業の開始点を明確に示すとともに，各手順についても詳細に示す（図 4.3 参照）．

(3) エラーメッセージとエラーの実例を，発生頻度が高いと思われる

「”（ダブルコーテーション）が足りない」

「スペリングの間違い」

の 2 例を加え，計 4 例に増やす（図 4.4 参照）．

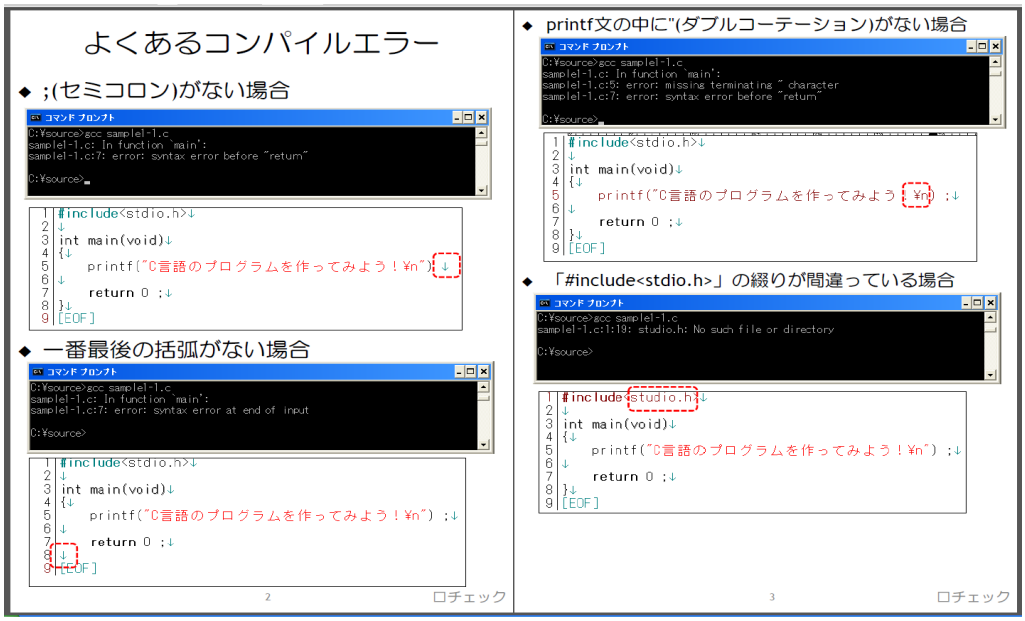

図 4.4: エラーメッセージの実例（2011 年度版）

4.5.2 改善した教材の実践における検証

2009 年度の授業では，コンパイルやエラー発見におけるつまずきに関しては提示した手順が分かりにくかったことや，一部の学生が手順を記したページを参照していなかったことから，さらに詳細な手順を別冊で用意して配布することで改善が可能であると考え，2011 年度の後期に京都大学において実施された同名の授業において，改善したテキスト教材を用いた実践と検証を行った．

対象科目とした科目と受講生の構成

本研究の対象とした授業は，2009 年度と同様に京都大学の全学共通教育科目「情報科学演習」であり，2011 年度の後期の授業を対象とした．受講生は 18 人で，学部別の内訳は，経済学部（６），工学部（４），農学部（３），総合人間学部（２），理学部（２），文学部（１）であった．また，プログラミングの既習者は４人であった．なお，2011 年度の授業内容,「確認テスト」および「最終テスト」は，2009 年度と同様のものを使用した．

2011 年度の授業実践におけるつまずき箇所と改善状況

2011 年度の授業では，全授業回を通して，受講生の９人から延べ 14 件のつまずきが確認されたが，とりわけ，2009 年度につまずき箇所として多く見られた「コンパイル手順」や「エラー対応」といった「実行までの手続きに係るつまずき」については，受講生数（初学者数）が，2009 年度の 15（13）人から，2011 年度の 18（14）人と若干ながら増加しているにもかかわらず，表 4.3 で示すとおり，つまずき箇所の件数はわずか４件と大幅に減少した．

テストの結果

2009 年度のテストでは,「繰り返し処理の for 文」に最も多くの誤答が見られ，確認テストで７人，最終テストで 11 人の誤答者があった．これに対して，2011 年度では，最終テストの際,「条件分岐処理の if 文」と「繰り返し処理の for 文」において各２人の誤答者が見られたのみであった．テスト結果の詳細を表 4.4 に示す．このように，プログラムを構成する基本的な概念の理解を測る問題，とりわけ「繰り返し処理の for 文」において誤答者数の著しい減少が見られた．

4.6　考察

4.6.1　自立的な作業遂行のための支援

本章では，プログラミングの学習において学習者の「自立的な作業遂行」と「作業を介した理解」に配慮した「写経型学習教材」を開発し，授業での運用を試みた．こうして開発した教材を用いたことに関し，2009 年度の授業実践では，開発したテキスト教材に「コンパイル手順」を示していたものの，これに係る「つまずき」が多数見られた．一方，2011 年度の実践では，コンパイルの手順やエラー発生時の対応など，実行までの手続きに係る「手順書」を作成し，別冊子で配布した結果，つまずき件数については 2009 年度の 38 件から大幅に減少し，４件となった．このような学習者に対する提示方法の工夫については，BRITTON ら [6] も教材の書き換え（記載順序の修正）や追記（見出しや下線の追記など）だけでも，教材の種類に関係なくおおむね学習の向上が認められた

と報告しており，わずかな変更であっても適切な修正あるいは改善であれば得られる効果は少なくないものと思われる．

4.6.2 「作業の自立的遂行」の改善による概念理解への影響

2011 年度の実践で使用した「手順書」は，直接的には，作業の自立的遂行の「つまずき」の軽減を目的に改善されており，同年度の実践では，意図したとおり，実行までの一連の過程におけるつまずきは大幅に減少している．加えて，この改善の効果は，作業におけるつまずきの減少にとどまらず，概念の理解を計るテストにおいても誤答者数の減少として顕在化しており，特に「繰り返しの for 文」のテスト結果において顕著である．

こうした結果は,「写経型学習」過程が for 文の概念の理解に一定の効果を持つことを示唆しており，一部限定的ながらも,「写経型学習」過程の必要性と具体的効果をプログラミング学習分野において初めて量的に示した事例となった．

阿部 [2] が「for 文の入れ子的な利用も初学者が典型的に困難さを抱えるものである」と指摘しているように，for 文の概念を理解させるためには相応の工夫が必要とされるが，本事例で用いたテキスト教材では for 文ドリルといった形で「写経型学習」過程の徹底を図っており，2011 年度の実践の際，テキスト本文と同ドリルに掲載されたプログラムの実行過程が滞りなく進んだことによって if 文と同程度の正答者数を示すまでに至ったと考えられる．

河内谷 [21] が指摘しているように,「代入という概念がすでに数学で学習済み」であるのに対して，for 文については「類似の概念を目にしたことがない」のだとすれば，学生は，類似性の高い既知の概念を見つけるのではなく，新たに概念を習得しなければならない．本実践では，異なる複数の事例から共通の要素を抽出して新たな概念を形成する，いわゆる帰納の過程によって，それを実現していたと考えられる．

4.6.3 本研究で用いた概念理解のための工夫とその効果

本研究で開発したテキスト教材では，当該概念を学ぶにあたり「何を目的として何をしているか」について予めテキスト教材の中で明確に説明しており，こうした方略により「作業を介した理解」の促進を期待している．この改善点が効果的に作用したか否かについては，授業実践の観察結果およびテスト結果な

どによって直接的に示されるデータはないが，少なくとも，授業中に見られた質問の中に「何を目的として何をしているかがわからない」といった類の質問は見られなかった．また，テスト結果を見ても 2011 年では，if 文と for 文の最終テストにおいてそれぞれわずか 2 例の誤答が見られたのみであることを考慮すれば，写経型学習過程にそうした工夫を加えるという簡便な方略だけでも，多くの学生が当該概念を理解するに至ったと解釈できる．一方，わずか 2 例であっても，理解できなかった学生も見られた．そのような学生には，ここで用いた「何を目的として何をしているか」について予めテキスト教材の中で明確に説明するといった方略だけでは十分でなかった，という可能性も否定できない．

4.7 本章のまとめ

本章では，第 2 章および第 3 章で述べた，企業研修の調査やその分析から得られた知見をもとに,「写経型学習教材」を開発し，同教材を京都大学における授業で用いて検証した結果を示した．

同教材については，2009 年度の後期の授業で初めて用いたところ，コンパイルの手順やエラー発生時の対応などにおける「つまずき」が多発したため，これらの問題点にいくつかの改善を加えており，2011 年度に再度授業実践を行っている．改善の結果，手順に係るつまずきが大幅に減少したほか，作業上のつまずきの減少に合わせて，特に，繰り返し処理の for 文の理解を問うテストにおいて誤答者数の大幅な減少が見られており，このように「写経型学習」を実施する際の手順の改善が学習課題そのものの理解に影響していることがわかる．これら 2 つの実践は,「自立的な作業の遂行」に係る非本質的な認知的負荷の排除によって当該学習課題の理解が大幅に改善されたことを示す事例である．

なお,「写経型学習」の過程から目的とする内容を学び取ることができないといった問題については，本実践で用いた改善が効果的に作用したか否かを直接的に示すデータは得られなかったが,「何を目的として何をしているかがわからない」といった類の質問は見られなかった上，テスト結果を見ても 2011 年度では，if 文と for 文の最終テストにおいてそれぞれわずか 2 例の誤答が見られたのみであった．このことを考慮すれば，写経型学習過程にそうした工夫を加えるという簡便な方略だけでも多くの学生が当該概念を理解するに至ったと解釈できる．

第5章　マイコンボード教材の活用による「作業を介した理解」の促進

5.1　研究の背景

コンピュータプログラミングの入門段階にある学習者が写経型学習過程を進める中でその作業を介して学習課題となる概念を理解していくわけであるが，ここには，大別して２つのつまずき要素が内在していると著者は考える．ひとつは，具体例の認知の困難性に伴うものであり，もうひとつは，当該学習課題そのものの認知の困難性に伴うつまずきである．

学習者は，サンプルプログラムを模倣する中で，しばしばサンプルプログラムそのもの，つまり，特定の処理を実現するための記述方法を暗記しているだけで，自らプログラムを作成する段階でそれを利用（応用）することができない [40]．このように具体的記述のみを記憶してしまう学習者が散見されるという事実は，この学習方法の中に，学習すべき機能や概念を理解しにくい構造的，認知的問題が潜んでいることを示唆している．この学習方法の手順は以下のようになる．

(1) サンプルプログラムに従って実際にプログラムを打ち込む
(2) プログラムをコンパイルし，実行する
(3) プログラムの実行結果を，個々の命令とそれらの実行結果（動作）の対応関係として理解する
の実行結果（動作）の対応関係として理解する．

このように，サンプルプログラムを用いたプログラミング学習は，事例から概念を学ぶ経験学習の様態を示している．このうち，(1) および (2) については，テキスト教材が使いやすかったり，あるいは教授者による指導が適切であれば，誰もが手順通りに進めることができる単純作業の過程であるが，他方，(3) については，「個々の動作を認知し，命令との対応関係として理解する」といった認知と思考の過程を含んでおり，この中に学習の困難性が潜んでいると考えられる．しかしながら，こうした困難性に対して克服を試みたいという内容の報告はほとんど見当たらず，教材を作成する際，一般に，十分な配慮がなされてい

るとも考え難い.

実際に用いられるC言語のサンプルプログラムには,「一つひとつの動作を確認する」学習過程には適していないプログラムがしばしば見られる．例えば，繰り返し処理を学ばせる過程で「****」のように「*」を複数出力させるというものがある．このプログラムの実行時のプロセスは，

「*」

「**」

「***」

「****」

と，実際には「*」がひとつずつ繰り返し画面上に出力されるものであるが，動作が一瞬にして終了してしまうため，学習者にとっては画面上に「****」と表示される最終結果しか確認できない．出版部数が多く，広く使用されているC言語のテキスト教材である「新版明解C言語入門編」[49]や「やさしいC」[55]などでもこのようなサンプルプログラムが採用されている．こうした動作は，視覚的に表出しているものの，個々の動作については認知可能な形で実行されていない.

つまり，こうしたプログラムでは，動作を確認しただけでは当該命令との対応関係を認識出来ないため，学習者に対し，潜在的な動作について推察する思考過程を要求することになる．推察や判別については，それを行うか否かを含め，学習者個人の習慣や類似する先行知識の有無，そして意思などを含めた総合的能力に依存するものであるし，加えて，推察や判別それ自体が認知的負荷となって学習者の認知資源（cognitive resource）を余計に消費することになる.

具体事例を認知して学習課題を理解する困難性に対しては，長谷川の試みのように，事前にイメージを形成させてから具体事例の認知を行う手法で対応したという報告事例は見られるが，この手法は認知の主体に対して働きかけるものであった．認知の対象に係る改善としては，プログラムの実行の可視化（例えば，[4]や[5]）のような提案はなされていたものの，それは単に内部的な処理過程をモニタ上に表出するいう試みであり，処理過程を認知させ，理解させる形態に改善するというところにまで踏み込んだ提案ではなかった．このように，認知の対象となる教材の作成に際しては，一般に十分な配慮がなされているとは考え難いのが現状である.

5.2 研究の目的

本章では，プログラミング学習の初期段階に見られる困難性の原因のひとつとして上述した問題，つまり,「命令の結果としての動作を視覚的に確かめることが困難な場合があること」，また「それが学習を阻害している可能性があること」を指摘するとともに，その改善方法として，プログラムの命令と動作の関係を視覚的に「顕在化」したカリキュラムおよび教材を開発た．また，これらを評価するため，2010 年度前期の京都大学の全学共通科目「情報科学演習」においてその評価を行った．

5.3 視覚的顕在化

5.3.1 概念理解に関する関連研究

概念の理解に関する研究

ある知識を記憶していることは確かだが使うことができないということはよくある．プログラミング学習の場合では，命令とその結果は記憶しているが，その関係を理解していない状態である．しかし，これでは概念学習を十分に達成しているとは言えないだろう．概念の学習水準は，概念の内容をいかに正確に検索できるかといった記憶の側面に加え，検索した概念をいかに広く問題解決に活用できるかという実行の側面からも評価されることになる [54].

一般に，知識には，それを学習した状況に制約されて，特定の目的や文脈と強く関連付けられた状態で獲得されるという性質（領域固有性）がある [16]．そのため，もとの学習を行った状況と新たな問題場面との間に，一見してわかる表面的な類似性や共通点がある，あるいは，先に学習したことが役立つことを教示するといった条件が満たされない限り，自発的に新たな状況に学習した概念を移転させることは困難になる [54].

このように，ある事例で学んだ概念を他の事例に応用する，いわゆる「概念移転」については，様々な報告がなされてきたが，その困難性の原因という視点でみると，概念の習得それ自体に注目したものと，知識とその周囲の概念との関係性に注目したものとの，おおよそ二つに分類できる．

前者としては，Kaminski らの報告をあげることができる．Kaminski らは，大

学生を対象として，具体例を用いて数学的な概念を学んだ学生と抽象概念から直接学び始めた学生を比較した．その結果，具体例で学んだ学生は新しい状況にその知識を応用できなかったのに対し，抽象的な記号などを用いて同じ概念を学んだ学生は，異なる状況に応用できる場合が多かった [20]．また，この報告では，結論の中で,「具体例を用いた学習では概念の抽出が困難であり，こうした学習から始めた場合，応用できないことがある」としており，抽象概念そのものの習得に重点をおいた方略を提起している．

一方，後者の概念間の関係性など周囲に注目したものとしては，Mayer などの報告がそれにあたり，概念移転の困難性を克服する方略として，学習する概念の体制化や統合，概念間の関係性の発見や先行知識との関連付けなどの認知的方略を使用する [32] と提案している．

これらの報告は，概念を習得し，その後，応用できるようになるまでの一連の過程の中で，各段階に対していかなる工夫や改善の方略が考えられるかについて示唆しており，相互に補完的である．これらの研究が示すように，プログラミング学習においても概念の習得と各概念間の関連付けは，概念移転を可能にする上で必要とされる要素であると考えられる．

プログラミングにおける概念の理解

プログラミング学習の分野に対象を絞ると，児童へのプログラミングの導入として Scratch というシステムを用いた実践では，児童が繰り返し処理や条件分岐処理を用いた作品を作成していたことから，プログラミングの基本的な概念が理解できたと結論づけている [36]．また，Papert[47] は，コンピュータがどのように考えるかを教えることを通じて，自分がどのように考えるかを探究できるとして，プログラミング言語 LOGO 上でタートル（亀）が動くグラフィックを用いた．これはその後，初学者への導入にしばしば用いられる手法となっている．さらに新開らは，プログラミングの導入教育として，プログラム作成プロセスを重視した学習支援システムを開発し，そのシステムを用いた C 言語の入門教育を実践した結果，プログラム作成プロセスに必要な知識と力の向上が見られている [51].

ただし，これらの研究は，学習支援のためのシステム開発といった側面が強

く，学習過程を分析的にとらえて各要素に対応した改善を試みてはいない．

先述したが，学習における理解の本質的課題について分析的になされた研究として，長谷川らの報告をあげることができる．長谷川らは，制御構造のイメージを自由に絵で表現する方法で，学習者のイメージを調査した結果，プログラミングを習得するには，プログラム実行時の処理の流れの明確なイメージを持っている学習者ほどプログラミングの理解度が高いことを示した [17]．さらに，長谷川らは，追跡調査を行い，初学者対象の授業の中で制御構造を表す図を提示し（図 3.1 参照），処理の流れのイメージを形成させることによってプログラミングの理解を支援でき，学習効果が得られることを明らかにした [18]．このように長谷川らは，抽象概念を直接学ぶ方法として「図」の活用を提案している．

5.3.2 概念理解のための「視覚的顕在化」

写経型学習では，記述（命令）が示す動作を確認する過程においてその学習目的を達成しようとするならば，動作は記述を反映したものとして認識されるよう明示されていることが望ましい．しかしながら，学習に使用されているサンプルプログラムの中には，当該命令の実行が動作として顕在化していないために，動作を確認しただけでは当該命令との対応関係を認識できないものが散見される．そこで，筆者は，命令と動作の関係を明確に把握するために，特に，出力としての動作を視覚的に「顕在化」することが有効であると考える．ここで,「視覚化」と呼ばず，あえて「視覚的顕在化」と表現した理由は，動作が網膜上に像が結ばれているといった物理的現象として「見える」ものだとしても，学習者が「認識して」いなければ，学習上の効果が期待できないからである．

つまり，動作は，学習者にとって「見える」だけでなく，それが認識しやすい状態でなければならない．すなわち，

- 大きさ，速さなどにおいて視認可能であること
- 周囲の視覚的要素と区別して認識できること
- 視認の主体が予測するあるいは容易に予測することのできる場所で予測する動作が実行されること
- 他の命令に基づく動作と区別，分離して視認できること

である．

表 5.1: プログラミング学習における視覚的顕在化

「視覚的顕在化」の 4 つの方法	改善前の問題点	視覚的に顕在な状態	視覚的顕在化による具体的解決方法
可視化	表示が小さい，動作が瞬時に実行されてしまう	1) 大きさ，速さなどにおいて視認可能な動作であること（視認性）	表示を大きくする，一つ一つの動作を視認可能な速さで実行されるようにする
識別容易化	動作が他の表示のなかに埋没してしまい認識しにくい	2) 周囲の視覚的要素と区別して認識できること（判別性）	動作の顕れる場所を分離するか，他の視覚的要素から際立たせる
予測可能化	どの場所でどのような動作が実行されているかわかりにくい	3) 視認の主体が予期するあるいは容易に予測することのできる場所で予測する動作が実行される（予測可能性）	実行される場所や動作を予め明示するか，あるいは，既存の知識や経験をもとに容易に予期できる動作の種類，動作の現れる場所に変更する
分離化	複数の処理の結果が区別しにくい一連の動作として顕れる	4) 他の命令に基づく動作と区別，分離して視認できること（独立性）	当該処理に対応する動作を一つに絞るか複数の動作をそれぞれ区別が可能なかたちに分離する

これらの要素を満たす教材の状態を本章では,「視覚的に顕在な状態」と呼ぶことにする．また，先の 4 つの状態へ移行されることを「可視化」,「識別容易化」,「予測可能化」,「分離化」の 4 つの側面によって実現を試みることとした（表 5.1 参照）．この 4 つの側面から構成されるプログラミング教材の改善について,「視覚的顕在化」と定義し，教材作成の際の指針として提案したい．

5.4 教材およびカリキュラムのデザイン

5.4.1 教材全体の設計方針

教材の設計では，前述した動作の「視覚的顕在化」に重きを置くとともに，概念を適用する複数の事例や例題を比較した [7] り，例題の解法や事例の説明を受容的に学ぶのではなく，学習者が自ら生成する [37] などの報告の知見を参考にして，以下の基本方針を定めた．

1. 視覚的に顕在な状態を確認できること（表 5.1 参照）
2. 複数の例題を使用すること
3. 学習者が自ら応用する時間を設定すること

5.4.2 開発したマイコンボード教材のデザイン

視覚的顕在化のうち，予測可能化は，テキスト教材に出力内容を明記すると同時にモニター画面上に出力に特化した領域を設けたり，自動車やロボットなどの既知の実物の模型を外部出力デバイスとして用いたりするなど，プログラム実行段階での出力の場所と態様について学習者が,「既に知っている状態」にしておくことが必要であり，また，その手段は複数想定される．本研究では，これらいくつか想定される手段の中から，学生にとって既知の形態として扇風機を模したマイコンボードを外部出力デバイスとして用いる手段を選択し，予測可能化の実現を試みることとした．また，扇風機を構成する各出力デバイスの数を 4 種類に限定し，かつ各デバイスの動作を単純化することで識別容易化が図られている．なお，マイコンボード自体の設計は A 社が担当し，その教材を用いたカリキュラムの開発は筆者が担当した．ここで開発された扇風機型マイコンボードは，マイクロプロセッサには H8 マイコンを採用しており，また，動作を実現するデバイスとして，視覚的に容易に把握でき単純な動作を示す「ライト (LED)」,「ファン（モーター）」,「スイッチ」と表示文字数が 16 文字と比較的少ない「LCD（液晶パネル）」の計 4 つを実装している（図 5.1 参照）．

プログラムの開発環境は，統合開発環境 GCC Developer Lite を採用した．受講生は，この統合開発環境を利用して C 言語のプログラムを作成する．プログラムが完成した後，クロスコンパイルを行い，マイコンボードに実行プログラムをダウンロードし，実際にプログラムを動作させる．また，マイコンボード

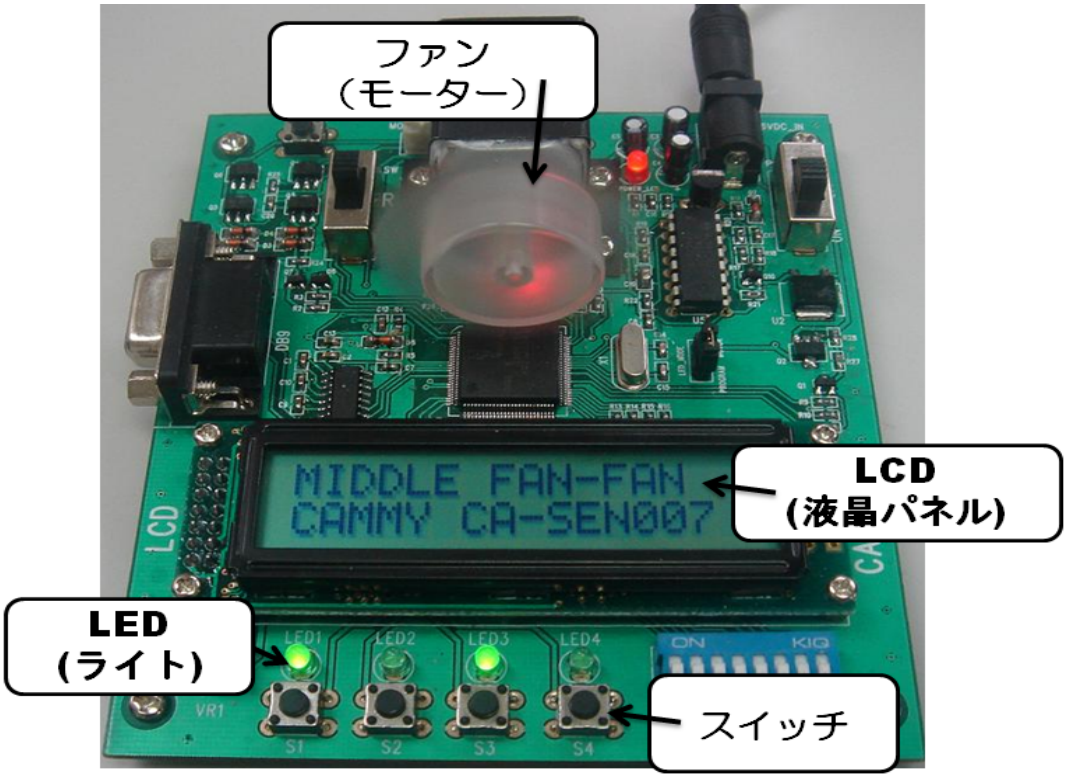

図 5.1: 使用したマイコンボード

上の入出力デバイスの操作は，学習者がプログラムを一から作成するのではなく，あらかじめ用意したライブラリ関数を呼び出す形式を採った．なお，マイコンの操作やマイコンボードで用いるライブラリ呼び出しの事例は，マイコンボードとあわせて開発したテキスト教材の中で紹介している．

5.4.3 カリキュラムのデザイン

本章では，5.3.2 で述べた特徴を踏まえて「視覚的顕在化」を意識したカリキュラムを設計した．テキスト教材の例題として，プログラムの命令と動作の関係を視認できるサンプルプログラムを数多く用意した（表 5.2 参照）．例えば，LCD に文字が時間差で表示されるものや 4 つのランプが時間差で点灯していくものである．学習の初期段階では，数行程度のサンプルプログラムを数多く用意し，1 つのサンプルプログラムで新規に学習する項目が多くならないよう配慮した．第一回目から第四回目までは，表 5.2 に示したとおり，視覚的顕在化に着目した教材とカリキュラムによって命令と動作の関係を明確に把握する段階として規

定した．とりわけ，第一回目および第二回目については，1 つの命令に対応した一つの動作を示すプログラムから，順を追って，最も多いものでも 3 つの連続した命令に対し 3 つの動作を示す比較的短いプログラムをサンプルとして提示している．このうち連続した命令と動作についてはステップごとに実行されるわけではないが，その直前に 1 対 1 の関係を学んでから行い，また，それらが極めて少数であることから，複数対複数の中にある個々の対応関係についても容易に推測が可能である．例えば，LCD の制御方法を学習する際，受講生は，文字の表示位置の指定方法を 8 つの例題から学んだあと，LCD 上に文字を複数出力させることにより，繰り返し処理を学習する．例えば，一秒ごとに「A」を表示させる命令を 16 回繰り返す．このとき LCD は，上段と下段にそれぞれ，

「AAAAAAAA」

「AAAAAAAA」

と計 16 個の「A」が表示される．その際，受講生は，文字が出力されていく過程を動作として視認し，処理過程と動作を関連付けながら学習を進める．

また，LCD 上に「A」を複数出力させる命令を一つずつ記述することによって順次処理も学習する．例えば，LCD の上段の左端に「A」を，左から 2 番目に「A」を，左から 3 番目に「A」を，4 番目に「A」を出力させる命令を記述する．このとき LCD は，上段の左端から 1 秒ごとに順次「A」を表示し，最後に「AAAA」と表示する．繰り返し処理同様，受講生は，動作を視認して処理過程と関連付けながら学習を進める．

次に，第三回目については，スイッチを用いた条件分岐を学ぶ段階であり，スイッチそれ自体は他のデバイスと組み合わせることによって機能するため，スイッチに対する命令と他のデバイスに対する命令と組み合わせたカリキュラムとなっている．これらの過程では，1 対 1 の関係をその都度確かめるためにステップごとに命令を実行する方法も考えられるが，本章では，これに替わる方法として，サンプルプログラムにおける各命令の実行過程を段階化するようにカリキュラムを構成し，学生が対応関係を類推することができるようにすることで個々の対応関係を把握できるよう試みている．

さらに，第四回目については，ファン（モーターとの連動）を用いて，今まで学習した命令（繰り返し処理と条件分岐）との組み合わせ方を復習する段階であり，ファンそれ自体は，スイッチとの組み合わせによって機能するため，第三回目と同様にファンに対する命令と他のデバイスに対する命令とを組み合わ

せたカリキュラムの構成となっている．

なお，授業の後半（第四回目以降）については，30 から 60 行程度のサンプルプログラムを用意しており，それまでに提示したものより長いサンプルプログラムになっている．その内容はそれまでに学習した一連のサンプルプログラムを組み合わせたものが主となっている．なお，テキスト教材のサンプルプログラムについては，実行可能な完全なソースコードとその実行結果を必ず示す形で用意した．また，テキスト教材では，コンパイルエラーの例示をするなどして学習者が主体的に学べるように配慮した．

5.5 授業実践

5.5.1 対象科目

本研究の対象とした授業は，京都大学の全学共通科目（全学部の 1 回生から 4 回生まで受講可）「情報科学演習」であり，2010 年度の春学期（4 月から 7 月まで）に実施した．本授業は，教養教育におけるコンピュータリテラシ教育の週 1 回 90 分（2 単位）の科目として実施した．なお，本授業のようにコンピュータリテラシの授業の中にプログラミング学習を取り入れる授業形態はあまり一般的ではないと思われるが，著者は，コンピュータがどのように動作しているか，また，何ができて何ができないかなどの基本的な事柄について実際のプログラミング体験をとおして理解を促そうと考えており，本授業はこうした方針の下，コンピュータリテラシ教育の一環として実践したものである．本授業の履修登録者数は 49 名であったが，そのうち，実際に授業に参加した学生は 35 名であった．

全体の授業内容は，Word による文書整形，PowerPoint と Excel の基本，HTML・CSS の基本についての授業を実施したのち，プログラミングの基本的な抽象概念の習得を目的として，マイコンボード教材を用いた C 言語でのプログラミングを 5 回取り上げた．授業の時間が限られていることから，学習内容は，変数と代入，および制御構造（順次処理，条件分岐処理，繰り返し処理）を中心とし，ライブラリ等の呼び出しを除いて関数は扱っていない．

授業では，5.4 で述べたように開発したマイコンボードとテキスト教材を学生一人ずつに提供し，一人 1 台のパソコンを用いて個人ごとに実習を行った．マ

表 5.2: 開発したカリキュラムの内容とその際に用いられた「視覚的顕在化」

回	学習内容	サンプル数	視覚的顕在化との関係
1	オリエンテーション（５回の授業の流れや開発環境および開発手順の 説明） ・LCD（液晶パネル）の制御（横16文字，縦２段まで表示可能） （１）LCDの一行目に文字列を表示させる． （２）（２）のプログラムを改変し表示位置をひとつ右へ移動させる ． （３）LCDの一行目および二行目に文字列を表示させる． な ど ・繰り返し処理 （４）for文を用いて文字の連続出力（LCDに１秒ごとに文字を右に表 示場所を移動させながら文字を出力する（１秒後は上段の左端，２秒後 は上段の２番目，３秒後は上段の４番目） ・順次処理 （５）文字の連続出力（繰り返し処理で用いた(1)の例題を順次処理で 記述したものを提示）	8 3 3	1)可視化：LCD上に視認に十分な大 きさ，１秒ごとという適度な速さで 文字が表示される 2)識別容易化：周囲に類似の表示 がない 3)予測可能化：LCD上という文字表示が唯一可能な場所で動作が実行さ れる 4)分離化：文字が１秒ごとに一つ ずつ表示される
2	第一回目の授業内容の復習や開発手順の確認後，第二回目の授業内容 の説明 ・繰り返し処理およびライトの制 御 （１）文字の処理と同様にwait処理をいれ，ライトが１秒おきに点灯 と消灯を５回繰り返す ・順次処理 （２）文字の処理と同様にwait処理をいれ，ライト１からライト４（ 左ライトから右ライト）を２秒ごとに点灯させる処理を順次処理で記 述 ・繰り返し処理 （３）（２）のプログラムをfor文で提示（実行結果が同じことを確認 させる） （４） while文を用いた永久ループ（処理を永久に繰り返す）処 理 （５）ディスプレイ，ライトの制御とfor文，while文を組み合わせたサンプル	3 2 5	1)可視化：LEDが視認に十分な１秒 ごとという適度な速さで点滅す る 2)識別容易化：周囲に類似の表示 がない 3)予測可能化：ライト上という「光 る」ことに特化した場所で限定さ れた「光る」という予測が容易な動 作が実行される 4)分離化：一つずつ点灯・消灯さ れる
3	・条件分岐とスイッチの制 御 （１）スイッチを押している間，押されたスイッチと同じ番号のライ トが点灯し（左端のスイッチを押したらその上にある左端のランプが 点灯），スイッチを離すとそのライトが消灯するも の （２）第２回目までに学習したLCDの制御とライトの制御を組み合わせ たサンプルを提示（例：スイッチ１が押下された場合，ランプ１を点 灯し，LCDの上段に“PUSH SW NO.1”と表示）	7	1)可視化：スイッチを視認に十分 な変化の大きさで上下させることが できる 2)識別容易化：スイッチと類似の 構造がほかにない 3)予測可能化：スイッチ上という 条件分岐を想像できる場所で作用す る 4)分離化：一つずつ動作される
4	・ファン（モーターと連動）の制御（繰り返し処理と条件分岐を組み 合わせたプログラムの学習） （１）スイッチ２を押下した場合，ライト２が点灯し，ファンが弱で 回転し，スイッチ１を押下した場合，ファンが停止するも の （２）スイッチ１を押下した場合，ファンを回転させ，ランプ１を点 灯させ，LCDに文字列を表示させ，スイッチ２を押下した場合，ファンを 停止させ，ランプ２を消灯させ，LCDに文字列を表示させるもの （３） 永久ループから抜ける仕組みとして， break文を用いたもの	4	1)可視化：ファンが視認に十分な 速度で回転する 2)識別容易化：ファンと類似の構 造がほかにない 3)予測可能化：ファン上という「回 転する」ことに特化した場所で限 定された「回転する」という予測が 容易な動作が実行される 4)分離化：ファンが回転する・停 止する
5	・「要求定義」に沿ったプログラムの作 成 （１）各スイッチの役割 ・スイッチ１：扇風機の電源ボタン ・スイッチ２：扇風機の弱ボタン ・スイッチ３：扇風機の中ボタン ・スイッチ４：扇風機の強ボタン （２）スイッチ１が押された場合，LCDに文字列を表示し，ライト１を 点灯する． （３）スイッチ２，スイッチ３，スイッチ４が押された場合， 扇風機の風の強さをあらわすメッセージを表示し，押下されたスイッチと同じ 番号のライトを点灯する．ただし，スイッチ１のライトは点灯したまま とする． （４）動作中にスイッチ１が押された場合，LCDにメッセージを約２秒間表示し，扇風機を停止，ライトをすべて消灯，LCDの文字をすべて消去する． 扇風機の動作として以下の条件を設定した． ・すべての動作はスイッチ１が押下されたあとに 有効となる ・風力は押されたスイッチにより変 更できる ・スイッチ1を押下することにより， ON／OFFの切り替えが何度でも可能	1	

イコンボードの貸出しは行わなかったので，プログラミングの実習は，授業時間中にのみ行える環境での授業である．授業開始時に PowerPoint の資料やテキスト教材をプロジェクタで示しながら要点のみを解説した．その後，受講生は

テキスト教材に従って，各々の学習進度に応じて学習を進めた．受講生がつまずいた場合は，教員やティーチングアシスタント（1 名）がサポートに入り，直接指導を実施したが，受講生の進度に合わせて統合開発環境やマイコンボードを示しながら解説を加える場合や口頭での説明も適宜行った．

5.5.2 データの収集・参与観察

本研究では，受講生のプログラミング経験を把握するため，初回の授業時にプログラミングの経験を問う質問紙調査を実施した．回答者数は 35 名であった．さらに実際の授業実践の現場で著者が参与観察を行い，

1. プログラムの命令と動作の関係の視覚的顕在化を題材としたプログラムの学習状況の把握
2. 複数の概念を組み合わせた課題の学習状況の把握

をするために，教員と受講生の対話の記録などを行った．さらに，学生の率直な意見をできるだけ多く集めるために，毎授業後にその日の授業の感想（自由記述アンケート）を担当教員に電子メールにて提出することを依頼した．これらが本研究の基礎データである．

5.5.3 受講生の構成

本実践での受講生は，35 名であった．その学部別の内訳は，総合人間（17 名），経済（6 名），法（4 名），文（4 名），教育（3 名），理（1 名）であった．また，プログラミング初学者が 29 名，プログラミングの経験があるものが 6 名であった．なお，本研究では，プログラミングの入門段階を対象としているため，「C 言語はすでに学んでいるが，組み込み系 C 言語にも興味がある」等の理由から参加したプログラミング経験者の 6 名については分析対象とせず，プログラミング初学者 29 名を対象に分析を行った．

5.6 実践結果

本実践は，学生らが対象を認知し，その中から学習すべき内容を抽出して身に付けていく経験学習の形態をとっており，「視覚的顕在化」は，学習のなかで対象の認知を容易にするための方略として位置付けられる．本節では，作成し

た教材およびカリキュラムが学習者の視覚的認知において有効に作用していたかについて，また，カリキュラムの妥当性について検証するため，観察された学習の経過と成果および授業後の感想メールの内容などを用いて，本実践における視覚的認知の状況や学習の進捗状況について示すこととする．

5.6.1 視覚的認知の状況

本実践では，授業後に自由記述式の感想メールを募っており，第一回目 28 名から，第二回目 27 名から，第三回目 27 名から，第四回目 27 名から，第五回目 28 名からそれぞれ各 1 件，延べ 137 名から総数 137 件のメールを受け取っている．

同メールに見られる記述は，受講生の主観的感想としての側面がある一方で，受講生が自由に選んで記述した内容であることから，「受講生がいかなるところに意識していて，また印象に残ったか」について客観的に把握する材料でもある．こうしたことから，本項では，教材が受講生の視覚的認知において有効に作用していたか否かについて検証するため，これらの記述の中から，特に「動作の視覚的認知と命令と動作の対応関係の把握」について受講生が意識していたことを示唆する箇所を抽出して以下に示す．

視覚的認知を介した動作と命令の関係の理解

第一回目の授業後の感想メールでは，「変数の指定によって文字列の表示される場所が移動したり，文字列が点滅したりするのがとてもおもしろかったです」など 2 件，第二回目では，「自分の指示したとおりにランプがついたり消えたりして面白かったです」「プログラムに対応してランプが光ったりするのは面白かったです」など 4 件，第三回目では，「自分でプログラミングしたとおりにランプがついたり文字が出たりするのはとても楽しかったです」との記述が見られた．これらの記述から，受講生らが，「自ら記述したプログラムの命令に基づいてその動作が実現すること」，つまり，「プログラムの命令と動作の対応関係」を意識していたことがわかる．こうした記述が全体を通して 7 例見られ，29 名の受講生のうち 7 名がいずれかの授業後の感想メールにおいてこうした記述をしていた（表 5.3 参照）．

表 5.3: 視覚的認知を介した動作と命令の関係の理解に関する記述

授業回	授業後の感想メールの記述
第一回	「プログラミングは初めてしました．少し間違えるとうまくいかなくなってしまって大変でしたが，自分の指示したとおりに文字が表示されるのが面白かったです」 「変数の指定によって文字列の表示される場所が移動したり，文字列が点滅したりするのがとてもおもしろかったです」
第二回	「自分の指示したとおりにランプがついたり消えたりして面白かったです」 「自分が指示したとおりに LED が点灯したり消えたりするのがとてもおもしろかったです」 「プログラムに対応してランプが光ったりするのは面白かったです」 「実際に LED が光るのでプログラムとの対応が分かりやすかった」
第三回	「自分でプログラミングしたとおりにランプがついたり文字が出たりするのはとても楽しかったです」

動作の視覚的認知

上述した「視覚的認知を介した動作と命令の関係の理解」のほか，第一回目の授業後の感想メールにおいて,「パソコン上での C 言語と組み込みのどちらを受講するか迷ったのですが，結果が視覚的にわかるのでこちらの方がいいと思います」「すべてパソコン上でやるよりも視覚的に分かり易かったです」との記述が見られ，出力の顕れる場所をマイコンボード上に設定したことにより，動作を視覚的に認知させることについて一定の効果があったことを示唆する記述も見られた．

動作の予測可能性

本実践では，第一回目に受講生から感想メールにおいて「自分が動きを想像しながらプログラムを記述して，その通りにうごくとうれしいです」，さらに，最終回である第五回目において「扇風機を使ったことがあるので，どういう風に動かしたいかは頭の中で想像できたので，どういう風にプログラムで記述すれ

表 5.4: 授業時間内にサンプルをすべて確認した人数および応用した人数

	第一回	第二回	第三回	第四回	第五回
学習内容	文字を表示させる	ライトを光らせる	スイッチを押す	ファンを回す	扇風機プログラムを作成する
サンプルプログラムをすべて確認した人数	15/28 名	24/27 名	27/27 名	27/27 名	–
応用した人数	3/28 名	7/27 名	14/27 名	20/27 名	28/28 名（全員完成）

ばよいのかを考えることに専念できた」との記述が 2 件見られた．わずか 2 名ではあるが，プログラムの動作をあらかじめ予測しながら学習を進めており，こうした点においても教材の視覚的特性が有効に作用していたと考えられる．なお，動作の予測可能性について直接言及するものではないが，これらの記述のほか,「プログラムでの記述の仕方がよくわかるので扇風機という題材で学べてよかった」など，扇風機という外部デバイスを用いたことについて肯定的な記述も見られた．

5.6.2 学習の進捗状況

本項では，参与観察および自由記述式感想メールの記述をもとに，学習の進捗状況について示す（表 5.4 参照）．

第一回目における学習の状況

ここでは，はじめに開発環境の操作方法や開発手順などの確認をしている．これら基本事項の確認の後，受講生らはサンプルプログラムを打ち込み始めたが，コンパイルエラーや実行ファイルをマイコンボードに転送する作業でつまずく者が見られたため，教員やティーチングアシスタントがサポートに入って問題を解決した．

続いて，受講生らは，教科書に記載されたサンプルプログラムをそのまま打ち込み，これにあわせて動作を確認していたが，授業時間内に 14 あるサンプルプログラムをすべて確認できていたのは，出席者 28 名の約半数である 15 名であり，これに対し，すべてを確認できなかった受講生は 13 名であった．作業が遅れた理由として，(1) タイピングに問題があり，サンプルプログラムの入力に時間がかかった，(2) 開発環境の操作に手間取った，が挙げられる．前者では，タイピングの速度が遅い受講生のほか,「#」など記号の入力方法がわからずに作業が中断してしまっている受講生もいた．後者では，プログラムの入力ミスによるコンパイルエラーを自身で解決できず，教員やティーチングアシスタントのサポートを待っていたことにより作業進捗に遅れが生じていた．さらに，すべてのサンプルプログラムは確認できなかった 13 名のうち 5 名は,「何が何だか分からぬまま，指示された通りのことをこなしていた感じでした」など理解に関してネガティブな内容の感想メールを記述していた．

第二回目における学習の状況

第一回目の授業において全てのサンプルプログラムを実行できなかった受講生は，残りのプログラムを実行した後，第二回目授業で提示した 10 のプログラムに取り掛かった．この授業終了までに，第一回目から第二回目までのサンプルプログラム計 20 をすべて実行，確認できていたのは出席者 27 名のうち 24 名である．

この 24 名のうち 8 名が，目的とする動作を自ら設定し，サンプルプログラムの応用を試み，7 名が応用に成功していた．このうちの一人は，授業後の感想メールで「教科書のプログラムを完成させた後，オリジナルでライトが左から右にすーっと流れていくプログラムを作りました」と記述している．こうした受講生の中からは,「教科書を見れば手順なんかも大体わかるので説明はもう少し簡略でもいいかもしれません」「もっとパソコンを触っている時間を増やして欲しいです」など，自分で作業する時間を増やして欲しいとの要望が出されていた．

また，初回の授業で理解に関してネガティブな内容の感想メールを記述していた受講生 5 名について見ると，3 名から「今日は前回よりも理解することがで

きました」など，二回目になって理解できるようになった旨の記述が見られた．なお，この 3 名は，第一回目から第二回目までのサンプルプログラムをすべて実行，確認できていた．二回目の感想メールにおいても「まだテキストに書かれていることが理解できないところがあります」「プログラミング難しいです」と理解に関してネガティブな記述をしていた受講生については，提示したサンプルプログラムを全ては実行していなかった．

第三回目における学習の状況

第三回目では，受講生は提示した 7 つのプログラムに取り組んだ．この授業終了までに出席者 27 名全員が第一回目から提示した総数 31 のサンプルプログラムをすべて実行，確認できていた．

出席者 27 名のうち 10 名が，これまで学習した範囲の中に理解が不十分なところがあり，不安を感じているなどの理由から，再度サンプルプログラムを打ち込み，動作を確認していた．

前述の 10 名のようにこれまでの学習内容の理解に不安を感じ，復習する受講生が見られた一方，27 名のうち 14 名については，これまでに学習した機能とプログラミングの概念を組み合わせて応用を試み，7 名がこれに成功している．こうした進捗度合いが比較的早いと思われる受講生は，授業後の感想メールで「プログラムの種類を知るとそれを組み合わせるのが楽しく思えてきた」「今までに習ったものと if 文を組み合わせて LCD 画面で視覚的に面白いことをしてみたいです」「今回はある程度条件分岐のやり方を覚えて，ものすごく長いオリジナルプログラムに挑戦していました（80 行くらい）．内容自体は単純で@の文字がボタンによって上下左右に動くというものです」など理解に関してポジティブな記述を残している．

なお，初回および第二回目の授業の感想メールで理解に関してネガティブな内容を記述していた 2 名から「授業の最初の頃はプログラムを写すだけでしたが，仕組みが徐々にわかってきました．自分で考えてプログラミングをするというところまではまだまだ至りませんが，練習を重ねていけば何とかなるのではと思えるようになっただけ進歩だと思います」「だいぶ慣れてきた気はするが，まだ自分で応用してプログラムを組めるかと聞かれると自信ないので，頑

張って理解したい」との記述が見られた.

第四回目における学習の状況

第四回目では，受講生は提示した 4 つのプログラムに取り組んだ．これらのサンプルプログラムは，これまで学習した繰り返し処理と条件分岐を組み合わせることによってファンと LCD，LED を連動させたものである．また，これまでのプログラムは長くても 30 行程度のものであったが，ここでは，60 行程度のサンプルプログラムを学習した．この授業終了までに出席者 27 名全員がプログラムをすべて実行，確認できていた.

この 27 名のうち 20 名が応用を試みてこれに成功していた．授業後の感想メールで「今日は，自分で考えてプログラムを変えて実行することができました．まだまだ理解できているとは言えない部分もたくさんありますが，だいぶ好きなようにいじることができたので楽しかったです」「今日はファンの使い方がわかったので，今まで学習した LCD や LED と組み合わせて自分なりにプログラムを作成することができた」「使える機能と命令が増えたので，自分でアレンジするのが楽しくなってきました．いろいろさわっているうちに，プログラムと動作の関係がわかってきました」などの記述が見られた.

また，同授業の後半において余裕のある受講生が増えてきたことから，第五回目に実施する予定であった扇風機プログラムに取り組む受講生も見られ，授業後の感想メールに「オリジナル要素が入っていますが，扇風機自体はもう完成しました」などと記述しており，この時点で既に受講生 3 名が扇風機プログラムを完成させていた.

第五回目における学習の状況

第五回目では，受講生は扇風機プログラムに取り組んだ．なお，ここでは，要求定義，つまり目標設定は教員側が行っているものの，完成例としてのサンプルプログラムは提示していない．こうした中，受講生は，それまで学んだ内容を応用して自らプログラムを作成し，出席者 28 名全員が扇風機プログラムを完成させていた．授業後の感想メールを見ると,「はじめていちから自分でプログ

ラミングをするようなものなので面白かったと思います」,「扇風機のプログラムを完成できたのでそれだけでこの授業の成果だなと思いました」など理解に関してポジティブな内容の記述のみが見られ，2 日目終了後に理解に関してネガティブな内容を記述していた 2 名についても,「今まではプログラムをただ写すだけという感じでしたが，今日の扇風機プログラムをやってみたら前よりは少し理解できました」,「プログラミングの理解はまあまあだと思います．最初のころは難しくて投げやりだったところもありましたが，諦めずにやれば案外理解できるものだなと実感できました」など，理解に関してポジティブな記述をしていた．

5.7 考察

5.7.1 視覚的顕在化と本実践における教材の有効性

本研究では，プログラムの命令と動作の関係の「視覚的顕在化」に着目した教材を開発し，授業での運用を試みている．こうして開発した教材を用いたことに関し，第一回目から第四回目の授業において，授業後の感想を求めたメールでは,「プログラムの命令と動作の対応関係」を学習する際に教材の視覚的認知性が有効であったことを示唆する記述が 29 名の受講生のうち 7 名から 7 件見られたほか，出力の顕れる場所をマイコンボード上に設定したことにより動作を効果的に認知できたことを示唆する記述や，さらに,「動作の予測可能性」に関わる記述がそれぞれ異なる学生 2 名から 2 件見られた．これらの合計は 11 名 11 件であり，こうした結果を踏まえると，開発した教材については,「動作の視覚的認知」,「視覚的認知を介した動作と命令の関係の理解」もしくは「動作の予測可能性」といった「視覚的顕在化」に関わる側面において，3 分の 1 以上の受講生に対しいずれかの側面で有効に作用していたと考えられる．

5.7.2 学習状況とカリキュラムおよび教材の評価

本実践において，第一回目では 28 名中 13 名，第二回目では 27 名中 3 名が教科書で提示したサンプルプログラムをすべては確認，実行できていなかったが，第三回目以降は出席者全員がこの作業を完了しており，この時点で感想メールに理解に関し，ネガティブな記述をしていた受講生もポジティブな記述をする

ように変わっている．

また，第四回目の授業後の感想メールに,「使える機能と命令が増えたので,自分でアレンジするのが楽しくなってきました．いろいろさわっているうちに,プログラムと動作の関係がわかってきました」との記述があることなどから，サンプルプログラムをただ写すだけでなく，それをアレンジしながら，プログラムと動作の関係を理解していく学生も見られているが，その一方で，第二回目の授業終了後に「教科書を見れば手順なんかも大体わかるので説明はもう少し簡略でもいいかもしれません」「もっとパソコンを触っている時間を増やして欲しいです」など，自分で確かめる時間が少ないという不満も見られた．

以上から考えると，本実践においては，一部の受講生にとっては，サンプルプログラムを実行する時間が十分でなく，また，自分でプログラムをアレンジしながら動作を確認する時間が不十分であり，本実践で用いたカリキュラムはこうした点において，改善の余地があると思われる．

また，第一回目と第二回目において，受講生がタイピングに慣れておらず，入力速度が十分でなかったのに加え，プログラミングで使用する記号の入力方法に関する知識の不足により，作業進捗に遅れが生じていた．こうした遅れが，サンプルプログラムをすべて試すことができないといった授業全体の遅れにつながっている．学習の初期段階でつまずかないためには，作業を遂行するために必要となるスキルや知識に対する支援も必要であり，用いたカリキュラムおよび教材は，こうした側面についても配慮が十分でなかったと考えられる．

一方，全受講生が自らの力で第 5 回授業の課題である扇風機プログラムを完成させており，目的とされる学習内容に関しては最終的に身に付けることが出来たと考えられることから，提示したサンプルプログラムの数，種類及び順序などを含む教材設計全体としては一定の妥当性が認められる．

5.8　本章のまとめ

本章では，具体例の認知の困難性に伴うつまずきに対処するため，動作を認知する際の推測や判別の要素をできるだけ排除することを試みた．ここでは，コンピュータの動作を認知可能な形にすること，とりわけ，視覚的に実行される動作に関して顕在化する方略を「視覚的顕在化」として提案し,「視覚的顕在化」という概念にしたがって，マイコンボードを活用することにより非本質的な認

知的負荷の排除を試みたカリキュラムを開発し，同教材を用いた実践を行った．その結果，受講生の 3 分の 1 以上から視覚的顕在化に関わる側面において効果があったと推察できる感想が寄せられた．

また，当該学習課題そのものの認知の困難性に伴うつまずきに対処するため，例えば，制御構造の条件分岐処理を学習する際に実物のスイッチを組み込んだマイコンボードを使用した．これは，実物のスイッチが条件分岐の概念を容易にイメージさせるとの考えから採用したもので，類似のイメージを提示することによって本質的な認知的負荷を軽減させるという試みであるが，受講生全員が課題を達成し，これらの教材とカリキュラムを用いた授業実践としても十分な成果を得た．

第6章　問題解決過程の模倣による学習のためのプログラミング教材の開発

6.1　研究の背景

第5章まで述べてきたように,「写経型学習」過程を経てプログラミングの基礎を習得したとしても，それだけでは，自ら新たなプログラムを構想し，そして作成する実践的なプログラミング能力を身に付けたとはいえない．和田 [56] が,「アルゴリズムや文法だけを教えるのであれば，個々のテーマごとに最適の例題や演習課題を用意するだけで，十分に対応することができる．市販されている教材の多くは，このような形式に基づいて記述されている．このような細切れの課題対応型に期待できるのは『コードの書き方』の修得までだろう」と言及しているように，実践的なプログラミング能力を身に付けるにはこうした学習だけでは十分でないとされている．これに対する改善方略については，Palumbo[11] が,「プログラミングのテキスト教材や教授法は，問題の明確化，要求分析，問題解決に関連した能力に重点を置くことが重要であり，プログラミング言語の学習から問題解決能力の養成へと転換すべきである」という旨の指摘をしているほか，複数の研究者が，大学の研究室での研究や実社会でのソフトウェア開発などで求められるプログラミングスキルを修得するためには，PBL を取り入れた演習科目（例えば，[44]）や OJT（On the Job Training）を取り入れた演習（例えば，[31]）により，先輩や教員の指導を受けながら，具体的な研究テーマや仕事に関連したプログラミング能力を獲得していくことが望ましいとの指摘がある．

こうした中，京都市立堀川高等学校では，1999 年から「総合的な学習の時間」と「情報」を包括する専門科目「探究基礎」（1 年次後期・2 年次前期）を導入して，基礎課程から応用過程までの一貫したプログラミング教育の実施を試みていたため，同校を対象として，その教育課程を調査，分析することとした．同校教諭の藤岡ら [13] は，この科目のうち，1 年次後期と 2 年次前期を対象として，プログラミング教育と PBL（問題解決学習）を有機的に統合したモデルカリキュラム ISEC-SET (Information Science Education Curriculum based on

PBL with Squeak Etoys) を開発し，2003 年から授業実践を続けてきた．

6.2 研究の目的

本章では，実践的なプログラミング能力の涵養を目的とした演習のためのカリキュラムおよび教材の開発を試みる．具体的には，高等学校を対象として，モデル化とシミュレーションにおける問題解決の過程を例示することを意図したエージェントベースシミュレーションを用いたプログラミング教材の開発を行う．また，同教材の評価をおこなうため，2009 年度の京都市立堀川高等学校「探究基礎」において，その評価を行った．

6.3 これまでの授業実践

6.3.1 ISEC-SET の授業内容

ISEC-SET の学習期間は 1 年間であり，問題解決プロセスをシミュレーションという題材を用いて教授することを目的としている．前半の 6 ヶ月は，オリジナル教材を用いてプログラミングの基礎（Squeak Etoys および Excel VBA）を習得する．後半の 6ヶ月は，生徒自らが研究テーマを設定して，コンピュータシミュレーションを用いて考察する個人研究を行い，研究過程や結果を論文にまとめるとともに，ポスター形式で発表を行う．

前半：プログラミングの基礎の習得

これまでの ISEC-SET の前半の内容を表 6.1 に示す．ここでは，プログラミングを用いた問題解決を体験しながら，プログラミングの基礎を習得する [14]．プログラミングの経験のない生徒でも，モデル化とシミュレーションの基礎を習得できる内容となっている．

後半：個人研究（生徒のテーマ設定による PBL）

ISEC-SET の後半では，生徒自身がテーマを設定し，個人研究を行う PBL を実施する．ここでは，教員はサポート役となり，適宜，面談やディスカッショ

表 6.1: 改善前の ISEC-SET 前半のテキストの内容

	学習目標	内容	時間
1	Squeak Etoys プログラミングの基礎	(1) Drive a Car プロジェクト：Squeak e Toyの基本操作の習得 (オブジェクト・ハロ・ビューワ・タイルプログラミング等)	2
2		(2) サイコロプロジェクト：アニメーションと乱数の扱い方	2
3	Squeak Etoysを用いたシミュレーションによる問題解決プロセスの体験	(3)モンテカルロシミュレーションによる円周率πの計算	2
4		(4) 斜方投射シミュレータの設計と実装	8
		(5) レポート作成	2
5	Excel VBAプログラミングの基礎	(6) VBAマクロ及びGUIの基本	2
6	Excel VBAを用いたシミュレーションによる問題解決プロセスの体験	(7) モンテカルロシミュレーションによる円周率πの計算	1
7		(8) 斜方投射シミュレータの設計と実装	2
		(9) 選択課題によるレポートの作成	2

ンを通して生徒のプロジェクト管理を行い，主体的な活動を支援する [14].

6.3.2 これまでの授業実践における問題点

ISEC-SET では，前半でプログラミングの基礎を学びながら，モデル化とシミュレーションの基礎を習得，後半で生徒自身がテーマを設定して個人研究を行う．しかしながら，これまでの実践では，毎年，後半の個人研究において，適切なテーマ設定ができずモチベーションが低下してしまう生徒が見受けられ，改善の必要性が示されていた．

授業の前半では，プログラミングの基礎の習得を目標としているが，ここでは，後半の個人研究において，生徒自らが研究テーマの設定を行うために必要となるモデル化とシミュレーションの考え方についても学習していたが，自らテーマ設定ができない生徒や自ら設定したテーマをモデル化できない生徒が散見されていた．その原因として，生徒たちは，前半の学習内容のどの部分を模倣すれば良いのかが理解できていなかったものと考察される．

PBL の実施において，生徒は各自が目標を設定し，それに向けてプログラムを作成していくわけであるが，目標設定から完成までの間には，解決しなければならない問題が多々ある．少なからずある課題のすべてに対し，生徒自身による問題解決を要求したり，あるいは教員によるファシリテートを必要とすると，授業時間や教員数だけを取ってみても授業として実現することは困難である．また，生徒に対して過剰な負荷を与えた場合，モチベーションを低下させてしまうことも懸念される．こうしたなかで，教員は，授業の中で学習者自身によって解決しなければらない問題の量を適切に設定する必要がある．そのため，本研究では，ひとつのシミュレーションモデルを模倣の対象として提示し，シミュレーションプログラムを一から考案するという問題解決の過程を省いているだけでなく，エージェントベースシミュレーションという型を指定することで，考えなければならないシミュレーションのパターンを限定することとした．

6.4 カリキュラム改善と授業内容

6.4.1 カリキュラムの提案

表 6.1 の従来のカリキュラムで実施していた「(9) 選択課題によるレポートの作成」では,「クラス会幹事があらかじめ用意すべき釣り銭枚数を求めるシミュレーション（以下,「つり銭シミュレーション」）」「かぜの感染シミュレーション」のいずれかを選択するものであった．後者の「かぜの感染シミュレーション」では，伝染病の感染モデルの基本である「SIR モデル」を採用していた．しかしながら，SIR モデルでは，かぜの感染状況を感染者数などの集計化した状態を用いて微分（差分）方程式で描写する．このモデルは，社会全体として考えた場合，どのように感染者数が増加し減少していくのかという過程を記述できるが，一方で，集計量の時間発展という抽象度の高い表現であるため，ある一人の感染者の経過過程や他人との関係を通じた社会全体への影響など現実世界との関連付けがしづらく，問題解決の過程がわかりにくい．本章では，これに代わるものとしてミクロな視点からの分析が可能なエージェントモデルの活用を提案し教材を開発する．

6.4.2　プログラミング教材の題材と実装

エージェントモデルとは，コンピュータ上に，エージェントと呼ばれる「擬人化」されたソフトウェアオブジェクトを多数作成し，仮想的な社会環境を与え，それらの動的な依存関係を観察するために人工的な社会を構築するものである [22]．エージェントを主体として考えることで，個体ごとのパラメータの違いや行動の違いなどをシミュレーションに組み込むことができる．これにより，より現実世界に近いモデルをより自然に構築できるため，生徒にとって理解しやすい内容となるものと期待できる [25].

6.4.3　カリキュラムの改善

ISEC-SET の後半で個人テーマ設定する際,
(1) 興味のあるテーマであること
(2) 自分で作業できること
(3) 半年で研究が終わる程度の難易度であること
という 3 点が重要となる．そこで，本章では，(2) の自分で作業できるようになることに重点を置き，前半の授業のカリキュラム改善および教材開発を行うことにした．開発する教材は，後半のテーマ設定の際の参考資料となるように問題解決の過程を例示することを意識した．また，(2) に重点を置くことで，生徒は研究の難易度のイメージを持つことができ，そのことによって研究対象の興味が広がることが期待できる．

新教材の開発にともない 2009 度にカリキュラムの改善を行った．その内容を表 6.2 に示す．Squeak Etoys によるプログラミングの基礎は，プログラミングの基本概念（順次処理，条件分岐処理，繰り返し処理）の理解を目的としていたため，大幅にコマ数を削減した．これに対し，Excel VBA によるプログラミングでは，シミュレーションによる問題解決プロセスの体験に時間を割くように改良を行い，モデル化とシミュレーションの理解に重点を置くことにした．そのため，表 6.1「(8) 斜方投射シミュレータの設計と実装」を削除し，従来は選択課題としていた「つり銭シミュレーション」と「かぜの感染シミュレーション」を全員が学習することとした．

表 6.2: 改善版 ISEC-SET 前半のテキストの内容

	学習目標	内容	時間
1	Squeak Etoys プログラミングの基礎	(1) Drive a Car プロジェクト：Squeak e Toyの基本操作の習得 (オブジェクト・ハロ・ビューワ・タイルプログラミング等)	2
2		(2) モンテカルロシミュレーションによる円周率 π の計算	2
3	Excel VBA プログラミングの基礎	(3) VBAマクロの基本	4
4		(4) モンテカルロシミュレーションによる円周率 π の計算	2
5		(5) クラス会幹事があらかじめ用意すべき釣り銭枚数を求めるシミュレーション	2
6		(6) レポートの作成	2
7	Excel VBAを用いたシミュレーションによる問題解決プロセスの体験	(7) かぜの感染シミュレーション	4
8		(8) レポートの作成	2

6.4.4 改善したカリキュラムを用いた授業デザイン

授業の流れ

開発した教材「かぜの感染シミュレーション（以下,「感染シミュレーション」)」を用いた授業は，表 6.2 の (7)，(8) に示したように計 6 コマで構成した．授業はプリントを用いて行った．特に，状態遷移図やフローチャートを作成する際は，生徒個人に考えさせながら，生徒の進度に応じて学習を進めていった．また，プログラムの実装にあたっては，過去に授業で扱ったサンプルを示し，生徒ができるだけ自分でプログラムを作成できるように導いた．なお，生徒が躓いた場合は，TA（１名）がサポートに入り，直接指導を実施した．

イントロダクション

授業は 6 コマ（1 コマ 50 分）の流れとテーマを紹介することから始めた．開発した教材のテーマは以下のものである．

テーマ：かぜの季節になると，他の人からうつされることがしばしばある．いま，クラスに 50 名の生徒が在籍しており，このうちかぜを発症しているのは１

人であるとする．このかぜは，発症者に接触することにより他の人にも伝染する．ただし，このかぜに一度かかると免疫ができ，同じかぜには感染しないものとする．このとき，クラス内でかぜが伝染していく様子をシミュレーションせよ．

モデルの提示と理解（1 コマ）

(1) 人の状態の変化を考える：
(a) 非免疫保持,
(b) 感染 (非感染源),
(c) 発症 (感染源),
(d) 免疫保持
(2) 非免疫保持から免疫保持までの状態の遷移を考える（状態遷移図で説明）
(3) 時間の概念について考える： 以下の条件を満たす条件遷移図を作成する．
条件：各生徒は，1 時間に 1 回，50 人の生徒の誰かとランダムに接触するとする（自分自身と接触することもあり得るとしている）．シミュレーションは 1 時間ごとに行う．感染してから発症するまでの時間は 72 時間（3 日間），発症してから免疫保持となるまでの時間は 168 時間（7 日間）とするとき，各生徒が各状態間を遷移していく様子を状態遷移図で表現せよ．

プログラム作成のために必要な部品の洗い出しと実装（1 コマ）

(4) 前節の状態遷移図をもとに，ある生徒が感染してから免疫保持に至るまでのフローチャートを作成する．
(5) シミュレーションに必要なパラメータを考える
(6) (4) のフローチャートをもとに，ある一人の生徒が感染後に状態を遷移していくプログラムを実装する．本論文では，Excel VBA の具体的なスクリプトは割愛する．

大規模なシミュレーションの実装（2 コマ）

(7) (4) をもとに 50 人でのシミュレーションを行うプログラムのフローチャートを作成する．

(8) (7) のフローチャートをもとに，50 人でのシミュレーションを行うプログラムを実装する．

結果からの考察（2 コマ）

(9) 感染シミュレーションのプログラムを完成させ，非免疫保持者数，感染者数，発症者数，免疫保持者数の変化を観察し，その結果を検証して，考察内容をレポートとして報告する．

6.5　授業実践

生徒によるアンケートの結果

新教材「感染シミュレーション」を用いた授業の評価を行うため，最終授業日に生徒に対し，アンケート調査を実施した．回答者数は，21 名であった．

「感染シミュレーション」の授業に関する評価を図 6.1 に示す．約 9 割の生徒が授業の内容について「難しかった」と回答しているが，一方で,「授業への興味」や「今後の学習への意欲」も同様に高い値を示している．このことから，学習内容は難しかったが，生徒の興味を喚起することができ，高い学習意欲を維持させたまま，次の学習（ISEC-SET の後半）につなげられたと考えられる．また,「モデル化の理解」および「シミュレーションの理解」については肯定的な評価が得られており，モデル化とシミュレーションの理解に重点を置くという授業の目的は達成されたと推察される．

次に,「感染シミュレーション」を完成させていく段階の難易度の評価を図 6.2 に示す.「かぜの感染というテーマそのものの理解」「人の状態を考えることおよびその変化を考えること」については,「易しかった」「やや易しかった」とする

回答が多く，難易度は低かったようである．一方,「1 人分の状態遷移をフローチャートで表現すること」および「プログラミングすること」に関しては,「難しかった」「やや難しかった」と回答する生徒が約 4 割いた．さらに,「50 人分の状態遷移をフローチャートで表現すること」および「プログラミングすること」に関しては,「難しかった」「やや難しかった」とする回答が約 8 割もあり，1 人から 50 人に人数が増えたことで，難易度が高くなったことが分かる．

教員の指導上の工夫

モデルの理解について，状態の検討や状態遷移図の作成では，できるだけ生徒個人に考えさせるようにした．教員は，生徒の意見を引き出し，それらを生徒に比較検討させながら一つのモデルを作成していくように導いた．次に，フローチャートの作成では，後のプログラムの実装を意識し，できるだけプログラムが書きやすい形になるように配慮して指導した．プログラムの実装は，教員がヒントを与えながらではあるが，できるだけ生徒が自分でプログラムを書けるように誘導した．その際，過去の授業で使用したテキストを参照しながら応用できる例を見せるようにした．また，単にプログラムを書かせるだけでなく，その論理的な理解ができているかをこまめにチェックしながら進めていった．これらにより，生徒は自分自身で考え，理解しながら作業することを通して少しずつではあるが自分で作業ができるようになった．

ISEC-SET 後半における生徒のテーマ設定

ISEC-SET の後半では，生徒全員が自らテーマ設定を行い，プログラム作成を行うことができた．その際，教員と TA は前半の学習内容を参照して指導を行った．

ISEC-SET 後半で個人研究を行うにあたり，どの授業が役に立ったのかを問うた結果（複数回答可）,「Squeak の講義と演習」を 5 人,「Excel VBA の講義と演習」を 14 人,「クラス幹事の釣銭シミュレーション」を 9 人,「感染シミュレーション」を 13 人,「レポート添削」を 17 人の生徒が挙げている．その中で,「かぜの感染シミュレーション」と回答した理由として,「段階的なレベルアップと

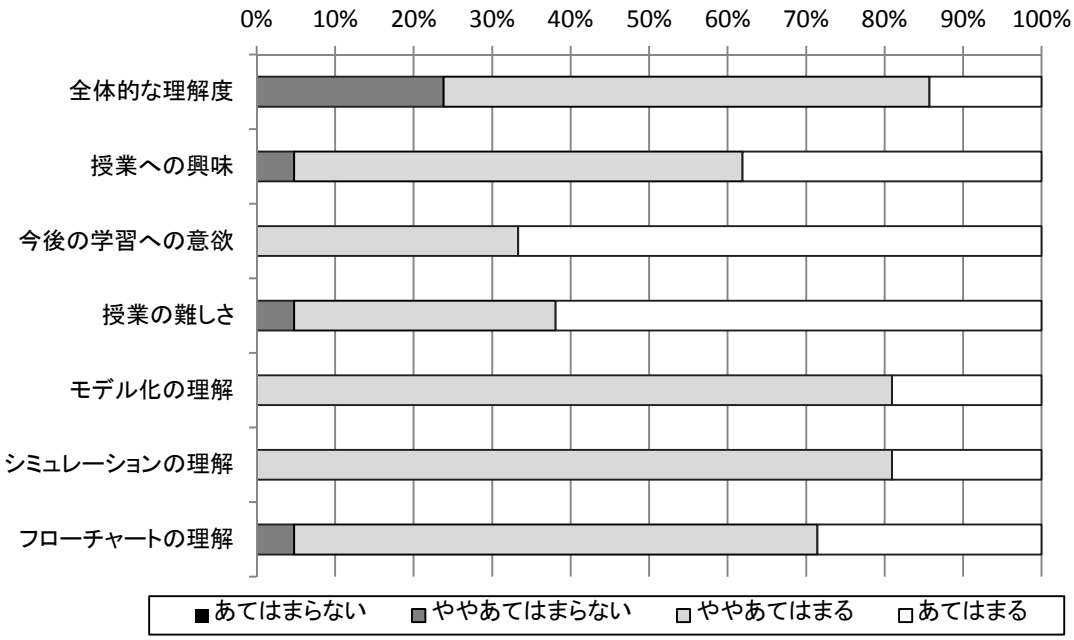

図 6.1: 感染シミュレーションの授業評価

必要な要素がしっかり盛り込まれたテーマのおかげで基本を修得できた」という意見があり，授業内容の難易度は高かったが，生徒は，個人研究を行うための基本を習得することができたと推察される．また,「プログラミングの基礎を習得することができたので自分がシミュレータを作成するにあたってそれらのスキルを応用できた」「実際にシミュレータを作成するときにどのように作っていけばよいのかを考えるのに役立った」という意見もあり，生徒は感染シミュレーションで学んだことを自分の研究に関連付けて考えることができたと解釈できる．2009 年度 ISEC-SET 後半で生徒が設定した研究テーマは,「感染シミュレーション」のモデリング方法を別の問題に置き換えるなど，授業で学習した内容を活用したテーマが数多く見受けられた．例えば,「棒引きでの優れた戦略を探る」「琵琶湖における外来種侵食シミュレーション」「客入りの良いお笑いライブとは」である．また，生徒がテーマ設定でつまずいた場合，教員は教材をもとに指導することができ，参考資料としての役割も果たしていた．これらのことから，集計量を用いる SIR モデルより個々の状態の変化や結果を理解しやすいエージェントモデルを用いた方が生徒にとって問題解決のプロセスやモデル化が分かりやすかったのではないかと推察される．

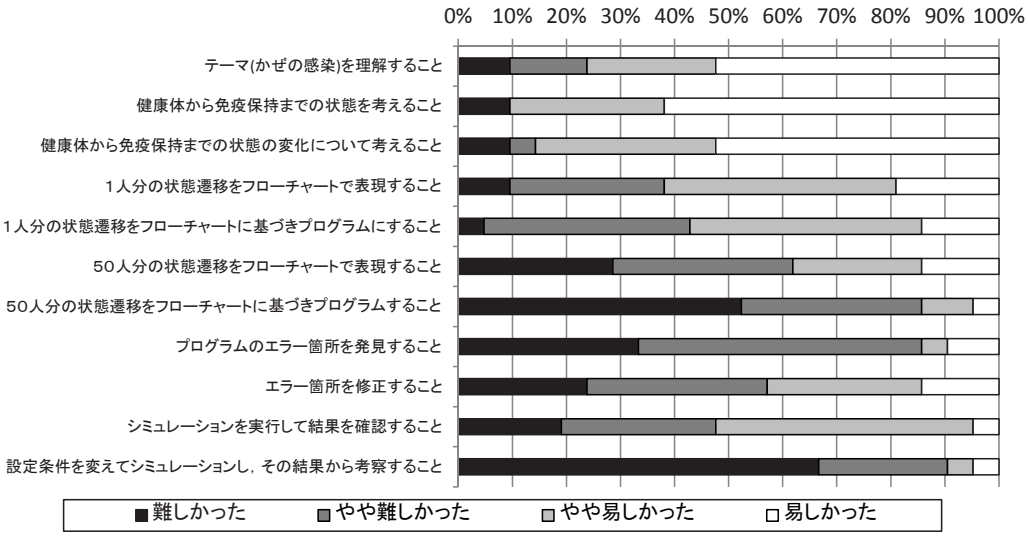

図 6.2: 感染シミュレーションに対する難易度

6.6 本章のまとめ

本章では，高等学校における問題解決型情報教育のためのエージェントベースシミュレーションを用いたプログラミング教材を開発した．本実践では，生徒に自身でエージェントの状態を考えながら，時間ごとにその状態を遷移させていくことを体験させることにより，モデル化とシミュレーションの意味を体験的に理解させることができた．同シュミレーションモデルを用いたカリキュラムは，現象，あるいは動態を個別の構成要素の行動から考えることが可能であり，自身を行動の主体と考える日常的な思考方法を流用できるため，集計量を用いるモデルを用いた場合に比べ，アルゴリズム構築の際に新たな概念の習得や複雑な思考を必要とすることが少なく，認知的負荷が低いものと著者は考える．応用力を涵養する段階においては，授業期間内に学習課題を達成できる範囲内で学習における負荷を設定する必要があり，同時に，学習者自身が考えて進めていく部分，あるいは問題解決していく部分も残しておかなければならない．こうした意味合いにおいて，本研究で開発したカリキュラムは一定の基準を達成できたものと評価できる．

第7章　おわりに

7.1　本研究のまとめ

プログラミングの基礎力の習得段階から，その応用段階にいたるまで，演習をいかに実施するか，さらにいかに支援するかといった課題は，プログラミング教育の効果的な実施を目標として包括的に取り組まなければならない課題であり，かつ，さらなる改善が求められる重要な課題である．本研究は，とくに演習における模倣のあり方に着目して，企業における研修の参与観察を伴う調査を行い，それをもとに大学や高等学校用の教材やカリキュラムの開発などから授業の改善を試みたものである．

本論文では，まず，学習者が入門段階の演習で，サンプルプログラムを模倣して入力，実行し，結果を確認することを重視して，これを「写経型学習」と規定し，この過程における学習に対する負荷の軽減を目的に，そのつまずき箇所の抽出と分析に基づく教材の開発を行った．続いて，実践的なプログラミング能力の涵養を目的とする学習段階における適度な支援のあり方について検討するため，PBL 型の授業における例示とその模倣に着目し，カリキュラムおよび教材を開発し，授業実践で使用することを通じて，この学習段階における支援のあり方について考察した．

第 2 章では,「写経型学習」過程におけるプログラミング初学者のつまずき箇所や問題点を調査し，初学者に見られる学習の困難性（潜在的なつまずきの要因）の抽出をした．それらは，概して「学習環境や学習者層などの条件によって生起する周辺問題」と「演習の中心課題としてのつまずき」の 2 つに大別できた．

第 3 章では，第 2 章で抽出した「演習の中心課題としてのつまずき」を，
(1)「作業の自立性」に係るつまずき
(2)「作業を介した理解」に係るつまずき
の 2 つに類型化した．さらに，先行知識の学習に及ぼす負荷を認知的負荷理論を用いて，
(1) 本質的な認知的負荷を伴う学習過程によるつまずき

(2) 非本質的な認知的負荷を伴う学習過程によるつまずき
の 2 つに区別し，それぞれに対応する学習方略を提示した．

第 4 章では，第 2 章および第 3 章で得られた知見をもとに「写経型学習教材」を開発し，同教材を京都大学における授業で用いて検証した．同教材を 2009 年度の授業で初めて用いたところ，コンパイルの手順やエラー発生時の対応における「つまずき」が多発したため，これらの問題点にいくつかの改善を加え，2011 年度に再度授業実践を行った．その結果，手順に係るつまずきが大幅に減少したほか，作業上のつまずきの減少に合わせて，特に，初学者が典型的に困難を抱える繰り返し処理についてその理解を問うテストにおいて誤答者数の大幅な減少が見られており，このように「写経型学習」を実施する際の手順の改善が学習課題そのものの理解に影響していることが分かった．これら 2 つの実践は,「作業の自立性」に係る非本質的認知的負荷の排除によって当該学習課題の理解が大幅に改善されたことを示す事例となった．

第 5 章では，具体例の認知の困難性に伴うつまずきに対処するため，動作を認知する際の推測や判別の要素をできるだけ排除することを試みた．ここでは，コンピュータの動作を認知可能な形にすること，とりわけ，視覚的に実行される動作に関して顕在化する方略を「視覚的顕在化」として提案し,「視覚的顕在化」という概念にしたがって，非本質的な認知的負荷の排除を試みるため，マイコンボードを活用した教材とカリキュラムを開発し，同教材を用いた実践を行った．また，当該学習課題そのものの認知の困難性に伴うつまずきに対処するため，たとえば，制御構造の条件分岐処理を学習する際に実物のスイッチを組み込んだマイコンボードを使用した．これは，実物のスイッチが条件分岐の概念を容易にイメージさせるとの考えから採用したもので，類似のイメージを提示することによって本質的な認知的負荷を軽減させる試みとしても位置付けられる．その結果，受講生の 3 分の 1 以上から視覚的顕在化に関わる側面において効果があったと推察できる感想が寄せられた．また，授業実践では，受講生全員が課題を達成し，これらの教材とカリキュラムを用いた授業実践としても十分な成果を得た．

第 6 章では，実践的なプログラミング能力の涵養を目的とした演習のためのカリキュラムおよび教材を開発し，実際の授業において用いている．具体的には，PBL 型の学習を行っている高等学校の授業を対象として，モデル化とシミュレーションにおける問題解決の過程を例示することを意図したエージェントベー

スシミュレーションを用いたプログラミング教材を開発した．同教材では，生徒自身が考えて進めていく部分，あるいは問題解決していく部分を設定されていたが，これを使用した授業実践では，全ての生徒がそれらの学習課題を授業期間内に達成することが出来た．

7.2 今後の展望

本研究は，プログラミング教育における演習過程の改善を包括的に取り組んだものであるが，まず，初学者を対象とした写経型学習過程の改善については，同過程のつまずきの分析と類型化を行い，教材開発や授業実践においてその評価を行った．しかしながら，実践機会の制約から提示した対応策に係る効果の検証についてはより一層の効果の把握が求められる．

もちろん，教育を対象とした研究である場合，その改善の試みについては，実用性を重視するとともに倫理的配慮も不可欠であることから，自然科学分野で要求されるような厳密な検証の必要性とのバランスが難しい．

例えば，本論文の第 5 章において,「視覚的顕在化」の方針のもとに実践したテキストおよびカリキュラムについては，その総体としては一定の効果が確認されたものの，方針そのものの効果については，それを実証するために十分なデータを収集できていなかった．仮に同様のカリキュラムを用いた場合であっても，学習者の振る舞いや各学習段階ごとの理解の程度を詳細に調べることによって，実証性の向上が見込まれると考えられる．今後は，これまでに行った改善については，検証方法等の工夫によってその実証性を高める必要がある．

これとは別の側面として，改善方略それ自体についても，より多く，またより効果的な提案が可能であると筆者は考える．本研究では，各つまずきを類型化したが，これらのつまずきの要因に対して，例えば,「作業を介した理解に係るつまずきにおいて当該学習課題それ自体の認知に困難性を伴う場合（本質的な認知的負荷）」については，課題の理解に適したプログラムあるいはそのプログラムの出力としての動作がいかなる態様をとるべきかについて，まだ改善の余地があると考えられる．今後は，こうしたつまずきに対応した具体的改善策，あるいは具体的な教材の作成を試みていきたい．

最後に，問題解決能力を涵養する過程については，高校生を対象としたカリキュラムおよび教材の作成を行い，実践で一定の効果は確認したものの，分析的

な取り組みではないため，提案自体が具体的である一方で一般性の高い提案とはなっていない．今後は，こうしたPBLの授業を，ソフトウェア開発で一般的な工程である要求定義やデザインの決定から，プログラミング（コード入力），そしてテストや評価までの各段階を分析的に調査することによって問題点あるいは改善可能な個所を浮き彫りにするとともに，その知見に基づいた教材開発を試みたいと考えている．

謝辞

本論文は，著者が京都大学大学院情報学研究科社会情報学専攻において，博士課程に在籍し，情報フルーエンシー教育分野（喜多研究室）において行ってきた研究を博士学位論文としてまとめたものです.

喜多一先生（京都大学国際高等教育院教授）には，平成 19 年 4 月に博士課程の学生として研究室に編入学して以来７年もの間，一貫してご指導をいただきました．私が，本研究に着手することになりましたのは，編入学して間もなく喜多先生より株式会社キヤミーとの共同研究にお誘いいただいたことに始まります．喜多先生からは，研究に対するアプローチや情報教育の捉え方など，学術的な面で薫陶を受けただけでなく，研究の周辺にさまざまな課題があってそれを乗り越えるための努力がいかに必要かを学ぶことができました．研究への直接的なご指導に加えて，研究のための環境やフィールドのご提供など，数え切れないご厚意をいただき，心より深く感謝申し上げます.

田中克己先生（京都大学大学院情報学研究科教授）には，博士課程編入学以来７年もの間，本研究のアドバイザや副査として多くのご指導をいただきました．田中先生からは，哲学的な思考法の身に付け方や研究の本質に立ち返ることの重要性についてご指導をいただき，情報学の観点からだけではなく，学問としての観点から多くのご示唆をいただきました．先生のご指導を通して,「研究とは何か」,「プログラミング教育とは何か」ということを深く考えることができるようになりました．いつも手厚いご指導をいただき，ここまで研究を続けることができましたこと，ここに深く感謝申し上げます.

守屋和幸先生（京都大学大学院情報学研究科教授）には，博士後期課程において，本研究のアドバイザや副査として多くのご指導をいただきました．本研究の捉え方や活用の方策などにもご意見いただき，私自身，本研究への新たな理解を産むことになりました．ここに深く感謝申し上げます.

酒井徹朗先生（京都大学名誉教授）には，博士課程において５年もの間，本研究のアドバイザや副査として多くのご指導をいただきました．博士の中間報告会にて，本研究に対し，励ましの言葉をいただいたときには，情報学研究科という場所で，教育の立場から研究している私にとって大きな勇気をいただきました．その後も手厚いご指導をいただき，ここまで研究を続けることができましたこと，心より感謝申し上げます.

高田秀志先生（立命館大学情報理工学部情報システム学科准教授）には，本研究のアドバイザとして多くのご指導をいただきました．本研究の立ち上げに際し，丁寧なご指導をいただきましたこと，深く感謝申し上げます．

村上正行先生（京都外国語大学准教授）には，教育工学というお立場から，本研究や論文執筆について多くのご指導，ご助言をいただきました．研究への直接的なご指導のほか，研究のための環境のご提供，非常勤講師などの職務を与えてくださるなど生活へのご配慮等のご厚意をいただきました．ここに深く感謝申し上げます．

株式会社キヤミーの吉川直人様には，研究のための環境のご提供のほか，教材の開発に関してご支援いただきました．吉川様とともに産学連携で研究できたこと，非常に貴重な機会をいただきましたこと，大変ありがたく思います．

また，先輩や後輩，研究に携わる仲間として支えていただいた，京都大学大学院情報学研究科社会情報学専攻情報フルーエンシー教育分野喜多研究室にこれまで所属してこられた諸先生方，研究員の方々，事務職員の方々，学生の皆様にも多くのご助言，ご支援をいただきました．皆様方に深く感謝いたします．

研究室の修了生である藤岡健史先生（京都市高等学校情報科・数学科教諭）には，研究のための環境のご提供いただいただけでなく，カリキュラムや教材開発，論文執筆の際に手厚いご指導，ご助言をいただきました．

研究室の事務補佐員である高田ひとみ様には，編入学してから 2011 年 8 月まで，種々の手続きにおいて大変お世話になりました．また，山下海華様には，2011 年 7 月から今日まで，種々の手続きはもちろんのこと，とりわけ，この度の学位申請において大変お世話になりました．

本研究は，喜多一先生が獲得された各種研究資金によってご支援いただいたのはもちろんのこと，2011 年度には，グローバル COE プログラム若手リーダーシップ養成プログラム研究費「学習目的に応じて組みかえ可能な組み込み系プログラミング教材の開発（研究代表者：岡本雅子）」による研究助成をいただきました．また，共立出版編集部の石井徹也様のご厚意により，本研究の成果を「写経型学習による C 言語プログラミングワークブック」（2012 年発行）としてまとめることができました．

最後になりましたが，私をあたたかく見守り，支えてくれた両親に深く感謝し，謝辞の言葉とさせていただきます．

平成 26 年 3 月　岡本雅子（京都大学吉田キャンパスにて）

参考文献

[1] A.Pears et al.: A survey of literature on the teaching of introductory programming, ACM SIGCSE Bulletin,Vol.39, Issue 4, 2007.

[2] 阿部圭一 ：C 言語によるプログラミング教育についての省察，情報処理学会研究報告「コンピュータと教育研究会」，No15，pp.205–212，2009.

[3] Anderson, J. R.，Cognitive psychology and its implications (2nd ed.), New York:Freeman，1985.

[4] Bharat Jayaraman, Charlotte M. Baltus：Visualizing Program Execution, Proceedings.,IEEE Symposium on Visual Languages, pp.30–37, 1996.

[5] Brad A. Myers: Visual Programming, Programming by Example, and Program Visualization: A Taxonomy, ACM SIGCHI Bulletin, pp.59–66, 1986.

[6] Britton, B. K., Gulogoz, S., and Glynn, S.：Impact of good and poor writing on learners: Research and theory，In B. K. Britton, A. Woodward and M. Binkley(Eds.), Learning from textbooks: Theory and Practice. New Jersey: Lawrence Erlbaum Associates, pp.1–46，1993.

[7] Catrambone, R., and Holyoak, K. J. ：Overcoming contextual limitations on problem solving transfer，Journal of Experimental Psychology，Vol.15，pp.1147–1156，1989.

[8] Chi, M. H., Bassok, M.：Learning from examples via self-explanations. In L. B. Resnick(Ed.), Knowing, learning and instruction，Essays in honor of Robert Glaser，pp. 251–282. Hillsdale. NJ: Lawrence Erlbaum Associates，1989.

[9] codeacademy, http://www.codecademy.com/(2014/01/26 参照).

[10] code.org, http://code.org/(2014/01/26 参照).

[11] D. Palumbo : Programming language/problem–solving research: a review of relevant issues, Review of Educational Research, Vol.60, No.1, pp.65–89, 1990.

[12] 江木鶴子, 竹内章:プログラミング初心者にトレースを指導するデバッグ支援システムの開発と評価，日本教育工学会論文誌, Vol.32，No.4，pp.369–381，2009.

[13] 藤岡健史，高田秀志，岩井原瑞穂：高等学校における Squeak を用いた課

題解決型情報教育の実践と評価. 日本教育工学会論文誌，Vol.28(Suppl.), pp.140–144, 2005.

[14] 藤岡健史：高校情報科学教育における「PBL 作品集」の可能性–カリキュラム ISEC-SET への導入とその効果–，平成 22 年度情報教育研究集会講演論文集, pp.407–410, 2010.

[15] 深町修一：文系の学生に対するコンピュータプログラミング教育の一考察，福岡国際大学紀要，Vol.23，pp.39–45，2010.

[16] Gick, M. L., and Holyoak, K. J. ：Analogical problem solving，Cognitive Psychology，Vol.12，pp.306–355，1980.

[17] 長谷川聡，山住富也：プログラミング教育と学習者のイメージ形成，名古屋文理短期大学紀要，Vol.22，pp.9–14，1997.

[18] 長谷川聡，山住富也：プログラミング教育と学習者のイメージ形成（その２），名古屋文理短期大学紀要，Vol23，pp.9–14，1998.

[19] 石田晴久訳：プログラミング言語 C　第 2 版，共立出版，東京，1989.

[20] Kaminski Jennifer A., Vladimir M. Sloutsky, and Andrew F. Heckler：The Advantage of Abstract Examples in Learning math，Science，No.320, pp.454–455，2008.

[21] 河内谷幸子：プログラミング言語の学習法–for 文の理解に関する認知心理学的分析–，言語と文化，法政大学言語・文化センター 編，No.3，pp.19–35，2006.

[22] 河合勝彦:認知的な制限を取り入れた新製品普及モデルの考察―エージェントベースモデルによるアプローチ―, オイコノミカ,Vol.44, No.2, pp.89–104，2007.

[23] 河西朝雄：なぞりがき C 言語学習ドリル–書き込んでいくことでしらずにしっかりマスター–，技術評論社，2008.

[24] 喜多一，岡本雅子， 藤岡健史，吉川直人：写経型学習による C 言語プログラミングワークブック，共立出版，東京，2012.

[25] 北中英明，高田朝子，横田絵理：ビジネス教育におけるエージェント・ベースト・アプローチの教育効果についての予備的一考察，経営行動科学,Vol.17, No.3, pp.159–172, 2004.

[26] 京都大学フィールド情報学研究会：フィールド情報学入門–自然観察，社会参加，イノベーションのための情報学–, 共立出版，東京，2009.

[27] Masako Okamoto, Hajime Kita, Isao Ono, Daisuke Kiga, Takao Terano, Takashi Yamada, Yusuke Koyama：Project-Based Learning of Computer Programming Using an Artificial Market System, ED-MEDIA CD-ROM, pp.4545–4553, 2008.

[28] Masako Okamoto, Mikihiko Mori, Hajime Kita, Isao Ono, Daisuke Kiga, Takao Terano, Takashi Yamada, Yusuke Koyama：Analysis of Self-Evaluation in Project-Based Learning of Object Oriented Programming, ED-MEDIA CD-ROM, pp.3016-3021, 2009.

[29] Masako Okamoto, Kayoko Terakawa, Masayuki Murakami, Kokolo Ikeda, Mikihiko Mori, Tetsutarou Uehara, Hajime Kita：Computer Programming Course Materials for Self-Learning Novices, In Procedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 2010), pp.2855–2861,2010.

[30] Masako Okamoto, Takashi Fujioka, Hajime Kita：Agent-based Modeling/Simulation for Teaching Science and Computer in High School, International Conference on Instrumentation, Control, Information Technology and System Integration (SICE 2011), pp.13–18, 2011

[31] 松永賢次：導入プログラミング教育におけるオンラインジャッジシステムの活用の試み，情報科学研究，No.31.pp.25–41，2010.

[32] Mayer, R.：Educational Psychology，A Cognitive Approach，Harper Collins，NY，1987.

[33] Mayer,R,E., Dow, G. T., and Mayer, S. : Multimedia learning in an interactive self-explaining environment: what works in the desgin of agent-based microworlds?, Journal of Educational Psychology, Vol.95, pp.806–813, 2003.

[34] 皆月昭則：失敗ソースコード学習と学生主体組織によるプログラミング授業に関する研究，鳴門教育大学情報教育ジャーナル，Vol.4, pp47–55, 2007.

[35] 文部科学省:新学習指導要領・生きる力，第２章　各教科　第８節 技術・家庭，http://www.mext.go.jp/a_menu/shotou/new–cs/youryou/chu/gika.html（2014/01/26 参照）.

[36] 森秀樹，杉澤学，張海，前迫孝憲：Scratch を用いた小学校プログラミング授業の実践–小学生を対象としたプログラミング教育の再考–，日本教育工学会論文誌，Vol.34，No.4，pp.387-394，2011.

[37] Needham, D. R., and Begg, I. M. ： Problem-oriented training promotes spontaneous analogical transfer，Memory-oriented training promotes memory for training, Memory & Cognition，Vol.19，pp.543-557，1991.

[38] 日本教育工学会編：教育工学事典，実教出版，2000.

[39] 岡本雅子，藤岡健史，喜多一：高等学校における問題解決型情報教育のためのエージェントベースシミュレーション教材の開発，Vol.35（Suppl.），pp.97-100，2011.

[40] 岡本雅子，村上正行，喜多一，吉川直人：初学者を対象とした自習中心のプログラミング教育の教材開発と評価，情報処理学会研究報告「情報教育シンポジム SSS2010 論文集」，pp.87–94，2010.

[41] 岡本雅子，村上正行，吉川直人，喜多一：「視覚的顕在化」に着目したプログラミング学習教材の開発と評価，日本教育工学会論文誌，Vol.37，No.1，pp.35–45，2013.

[42] 岡本雅子，村上正行，吉川直人，喜多一： プログラミングの写経型学習過程を対象としたつまずきの分析とテキスト教材の改善–作業の自立的遂行と作業を介した理解のための支援と工夫–，京都大学高等教育研究，No.19，pp.47–57, 2013.

[43] 岡本雅子，喜多一：プログラミングの「写経型学習」における初学者のつまずきの類型化とその考察，パイデイア，滋賀大学教育実践研究指導センター紀要，No.22，pp.49–53, 2014.

[44] 小野功ほか：徳島大学工学部知能情報工学科における創成型科目への取り組みーネットワークを用いた対戦型ゲームのグループによる作成ー，平成 12 年度工学・工業教育研究講演会講演論文集，pp.325-326，2000.

[45] 太田信宏：Java プログラミング教育に関する一考察，立教大学女子短期大学部研究紀要，Vol.52，pp.1-16，2009.

[46] 大島純:最近の認知研究からみた e-ラーニングの可能性, The Annual Report of Educational Psychology in Japan，No.47，pp.178–187，2008.

[47] Papert,S.：Mindstorms: Children, Computers, and Powerful Ideas. Basic Boks, NY，1980.

[48] Pass, F., Tuovinen, J. E., Tabbers, H., Vab Gerven, P. W. M.：Cognitive load measurement as a means to advance cognitive load theory，Educationl Psychologist，No.38，pp.63–71，2003.

[49] 柴田望洋：新版 明解C言語入門編，ソフトバンククリエイティブ，東京，2004.

[50] 柴田庄一，遠山仁美：技能の習得過程と身体知の獲得–主体的関与の意義と「わざ言語」の機能–名古屋大学言語文化論集，Vol.24, No.2, pp.77–93, 2003.

[51] 新開純子，宮地功：プログラミング学習支援システムを用いた入門教育の実践，日本教育工学会論文誌，Vol.33（Suppl.），pp.5-8，2009.

[52] 情報処理学会：情報処理, Vol.52，No.8，pp.911-981，2011.

[53] Sweller, J., van Merrienboer, J.J.G., Pass, F.：Cognitive architecture and instructional design，Educational Psychology Review，Vol.10，pp.251–296，1998.

[54] 高垣マユミ：授業デザインの最前線　ー理論と実践を創造する知のプロセスー，北大路書房，京都，2010.

[55] 高橋麻奈：やさしいC 第4版，ソフトバンククリエイティブ，東京，2012.

[56] 和田浩：スパイラル・アプローチによるC 言語プログラミング教育の実践と評価–受講生中心の学習を目指した試み–，UNISYS TECHNOLOGY REVIEW, Vol.62, pp.104–119, 1999.

主要業績

書籍

1. 喜多一，岡本雅子，藤岡健史，吉川直人．写経型学習によるC言語プログラミングワークブック，共立出版，東京, 2012（第４章に該当）

学術論文誌

1. 岡本雅子, 藤岡健史，喜多一：高等学校における問題解決型情報教育のためのエージェントベースシミュレーション教材の開発, 日本教育工学会論文誌 35 巻 Suppl., pp.97-100, 2011（第６章に該当）
2. 岡本雅子，村上正行，吉川直人，喜多一：「視覚的顕在化」に着目したプログラミング学習教材の開発と評価，日本教育工学会論文誌，Vol.37, No.1, pp.35–45，2013（第５章に該当）
3. 岡本雅子，村上正行，吉川直人，喜多一：プログラミングの写経型学習過程を対象としたつまずきの分析とテキスト教材の改善－作業の自立的遂行と作業を介した理解のための支援と工夫－，京都大学高等教育研究，No.19, pp.47–57, 2013（第４章に該当）
4. 岡本雅子，喜多一：プログラミングの「写経型学習」における初学者のつまずきの類型化とその考察．パイデイア，滋賀大学教育実践研究指導センター紀要，No.22，pp.49–53, 2014（第２章に該当）

国際会議

1. Masako Okamoto, Kayoko Terakawa, Masayuki Murakami, Kokolo Ikeda, Mikihiko Mori, Tetsutarou Uehara, Hajime Kita：Computer Programming Course Materials for Self-Learning Novices, In Procedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 2010), pp.2855–2861，2010 （第４章に該当）
2. Masako Okamoto, Takashi Fujioka, Hajime Kita：Agent-based Modeling/Simulation for Teaching Science and Computer in High School, International Conference on Instrumentation, Control, Information Technology and System In-

tegration (SICE 2011), Tokyo, Japan, Sep. pp.13—18, 2011（第 6 章に該当）

口頭発表（査読有り）

1. 岡本雅子，村上正行，喜多一，吉川直人：初学者を対象とした自習中心のプログラミング教育の教材開発と評価，情報教育シンポジム SSS2010 論文集, pp.87-94，2010 （第 2 章に該当）